UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

EXPLAINABLE ARTIFICIAL INTELLIGENCE - LEARNING DECISION SETS WITH SAT

Filipe Inácio da Costa Pereira

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Interação e Conhecimento

Dissertação orientada por:
Prof. Doutor João Paulo Marques da Silva

2018

Acknowledgments

My first acknowledgement will go to my advisor João Marques Silva, not only for being an expert in the fields of Logic but also because he never hesitated to help or lend an ear. Without him, I wouldn't have as much of a good time as I had doing this. Thanks to him, I was also able to attend the Federated Logic Conference 2018, in Oxford. As he always said, *"Plans are worthless, but planning is everything"*. Also invaluable was Alexey Ignatiev, his early tips gave me the head start necessary to dive in the world of logic and research. I would also like to thank United Technologies Research Center for my research grant.

To my family, my mother, Maria Isabel, who sacrificed herself so I could be who I want, to my father, Joaquim Pereira, who paid my university fees, to my brother, Tiago Pereira, who was always there and finally, my two beautiful nephews, Miguel and Leonardo.

To the people that helped me with the timeline of my thesis. Specifically, in delaying it. Procrastination did wonders. To the people that have been present since my high school, my group "Spiders", Sebastião Peixoto, Rafael Pacheco, Bernardo Pina, Henrique Vale, Tiago Ferreira, Pedro Bengaló, Pedro Pereira and João Pereira.

To the faculty group, "Questões da Vida", João Antunes, João Cardoso, João Batista, João Rodrigues and Dharmite Prabhudas.

Finally, to the grammar police, Margarida Penedos, who LOVED my use of commas and once again, João Batista.

I think that's enough sissy stuff. "Bla bla bla"and to everyone, once again, thank you.

To everyone that made this journey possible.

Resumo

A Inteligência Artificial é um tema de investigação fulcral no crescimento tecnológico e com um forte impacto desde a sua criação na conferência de Darmouth, em 1956, onde foi proposta a seguinte asserção sobre a inteligência artificial: "Todos os aspetos da aprendizagem ou qualquer outra caraterística de inteligência pode ser tão precisamente descrita que uma máquina pode ser feita para a simular". Com o constante aumento de informação (*Big Data*) ao longo dos anos, temos modelos cada vez mais eficientes que em poucos segundos nos informam da sua predição sobre um dado conjunto de *input*, dos quais se destacam os *Black Box Models*, sendo estas as técnicas com melhores resultados e com maior complexidade. Infelizmente, estes não nos conseguem fornecer uma explicação para a sua predição, o que pode ser uma enorme desvantagem para nós seres humanos. *Explainable Artificial Intelligence*, que visa associar explicações a decisões feitas por agentes autónomos, quebra esta falta de transparência. Visto que para a tomada de decisões é necessário mais que uma etiqueta, será bastante útil uma explicação, criando assim uma relação de confiança com o modelo implementado.

Irão ser apresentados conceitos preliminares tais como o problema de satisfação, o problema da classificação, complexidade, problemas MaxSAT, subconjuntos de satisfiabilidade máximos e subconjuntos de correção mínimos. Um dos objetivos é apresentar uma base de conhecimento sobre *Explainable Artificial Intelligence*, que relaciona uma forte componente de vários modelos e *frameworks* e se encontra dividida em duas grandes componentes: a de criação de modelos que já são interpretáveis por si ou a de criação de *frameworks* que justificam e interpretam predições escolhidas de quaisquer modelos. Os modelos interpretáveis podem ter uma abordagem heurística ou exata (com lógica). Dentro destes modelos, são explorados mais a fundo os Conjuntos de Decisão, sendo estes os mesmos em que o nosso trabalho se baseia e comparada a sua eficácia com a nossa abordagem

As *frameworks* baseadas na interpretação e justificação de predições são também muito importantes pois permitem a justificação de qualquer predição de um modelo, de forma fiável e entendível por seres humanos. A maioria deste trabalho é focado em redes neuronais, sendo estes os modelos com melhor precisão mas sem justificação associada a cada predição. Também é estudado o problema de minimização de árvores de decisões. Este paper refere um modelo que, usando uma árvore fixa dada, consegue minimizar o

número de nós. Um dos problemas a notar é uma grande complexidade de espaço que não permitiu resultados fiáveis ou a existência dos mesmos. Os papers "Interpretable Decision Sets" e "Minimizing Decision Tree Size as Combinatorial Optimization" foram a base da abordagem de lógica no mundo de Aprendizagem Automática.

Nesta tese será feita a descrição e análise da implementação de dois modelos interpretáveis (Conjunto de Decisão e Árvores de Decisão) baseados em lógica, através de Solucionadores SAT (Satisfiability) ou SMT (Satisfiability Modulo Theories). Este trabalho foi motivado por uma análise em detalhe de trabalho na área de Inteligência Artificial Interpretável, com o propósito de procurar aplicações de lógica neste domínio.

Os Conjuntos de Decisão têm tido um progresso formidável nestes últimos anos e conseguem providenciar explicações sucintas e precisas. Estes são conjuntos de declarações "If-then". Uma regra é constituída por um tuplo (π, c), em que π é um conjunto de itens e c a classe atribuída, e é interpretada da seguinte maneira: SE os literais em π forem verdadeiros, ENTÃO escolha a classe c. Um exemplo duma regra é: "Vacation:1 $\rightarrow$ Hike: 1". Existem listas/conjuntos de decisão (que impõe ordem e estão conectados por declarações "else") e listas/conjuntos de decisão (que não necessitam de ordem). Apesar de à partida os últimos aparentarem ser mais interpretáveis, com eles estão relacionados problemas tais como a sobreposição de regras.

A ferramenta desenvolvida é denominada de MINDS e a nossa abordagem de Conjuntos de Decisão começa com uma definição de conceitos importantes, baseados em trabalho anterior [1]. Conceitos como *itemsets*, regras, regras de definição e sobreposição de regras (quer seja no *dataset* ou no espaço de atributos) são explicados. A geração de explicações sucintas também é esclarecida. A nossa abordagem requer que os *datasets* estejam devidamente binarizados e considerando que maioria dos *datasets* não se encontra assim, serão aplicadas técnicas de "one-hot encoding". A binarização de um *dataset* implica transformar atributos categóricos em vetores binários mas esta aumenta ligeiramente a complexidade de espaço do nosso modelo devido ao aumento de atributos. Os vários problemas de otimização do MINDS são abordados e apresentados a explicação da sua complexidade, tais como as variáveis e restrições de cada modelo. Cada modelo terá também um exemplo de um Conjunto de Decisões. A aprendizagem de Conjuntos de Decisão foca-se no *dataset* e o objetivo é passar através do mesmo, criando variáveis e restrições e atribuindo valores às mesmas com um solucionador SAT/SMT, permitindo a minimização do número de regras (ou Formas Normais Disjuntivas) para cada representação de classe binária (c_0 e c_1) e evitando sobreposição nas regras, sendo estas pelo *dataset* ou pelo espaço de atributos, mantendo assim explicações interpretáveis e com precisão perfeita. Trabalho anterior [1] focou-se apenas em sobreposição de regras no *dataset*, na nossa abordagem existe o foco em retirar a sobreposição de regras no espaço de atributos. Também é realizada uma abordagem de quebra de simetrias (pois existe uma falta de ordem nas regras), de forma a melhorar o espaço das restrições e obter

mais desempenho. Impor uma ordem implica uma redução na complexidade e não afeta a precisão.

Para Árvores de Decisão existem abordagens práticas realizadas com procura heurística para a aprendizagem das mesmas. A nossa abordagem de Árvores de Decisão com SAT, cria restrições que codificam uma árvore binária válida e, por sua vez, restrições que garantam que a árvore de decisão seja perfeita (100% precisa) enquanto classifica os exemplos no *dataset*. Se as restrições forem válidas quando passadas por um solucionador SAT, este irá retornar uma árvore de decisão válida e otimizada. Tal como na abordagem anterior, também é usada uma maneira de quebrar simetrias aqui: ao assegurar que aos ramos da esquerda sejam concedidos um atributo com o valor 0 e aos ramos da direita com o valor 1. Este modelo desenvolvido tem uma complexidade muito inferior ao do paper "Minimizing Decision Tree Size as Combinatorial Optimization"e não requer uma árvore de decisão fixa.

Houve também investigação extensiva na comparação dos modelos referidos, tais como o IDS, JRIP, PRISM, CN2 e modelos do MINDS. Uma grande parte dos *datasets* usados nos resultados experimentais eram inconsistentes, tendo várias ocorrências do mesmo exemplo com classes atribuídas diferentes. A forma de resolver tal situação foi atribuir a classe C ao subconjunto maior. Os nossos modelos necessitam de *datasets* consistentes, senão as restrições nunca seriam solucionadas (pois seria impossível chegar a 100% de precisão). Os testes foram realizados em 49 datasets e impostos um tempo limite de 600 segundos. Estes datasets tinham uma grande quantidade de linhas, atributos e valores distinctos em cada atributo. Os modelos SAT têm como objetivo minimizar o número de regras mas de forma a tornar os conjuntos de decisões mais concisos, foi realizada uma técnica de enumeração de subconjuntos de correção mínimos. A avaliação foi feita nos 49 *datasets*, baseando os critérios no número de regras e literais, precisão e tempo. Existe uma pequena discussão das novas Regulações Europeias de Proteção de Dados, pois estas legislações criam o direito de ter uma explicação e restringem perfilagem. Concluindo, existe uma visão geral positiva devido aos resultados experimentais obtidos, possivelmente com um objetivo futuro de tornar os modelos mais eficientes e com menos literais de forma a competir com outros modelos.

Palavras-chave: Inteligência Artificial Explicável, Modelos Interpretáveis,
Interpretação e Justificação de Predição, Conjunto de Decisão, Solucionadores SAT.

Abstract

Artificial Intelligence is a core research topic with key significance in technological growth. With the increase of data, we have more efficient models that in a few seconds will inform us of their prediction on a given input set. The more complex techniques nowadays with better results are Black Box Models. Unfortunately, these can't provide an explanation behind their prediction, which is a major drawback for us humans. Explainable Artificial Intelligence, whose objective is to associate explanations with decisions made by autonomous agents, breaks this lack of transparency. This can be done by two approaches, either by creating models that are interpretable by themselves or by creating frameworks that justify and interpret any prediction made by any given model.

This thesis describes the implementation of two interpretable models (Decision Sets and Decision Trees) based on Logic Reasoners, either SAT (Satisfiability) or SMT (Satisfiability Modulo Theories) solvers. This work was motivated by an in-depth analysis of past work in the area of Explainable Artificial Intelligence, with the purpose of seeking applications of logic in this domain.

The Decision Sets approach focuses on the training data, as does any other model, and encoding the variables and constraints as a CNF (Conjuctive Normal Form) formula which can then be solved by a SAT/SMT oracle. This approach focuses on minimizing the number of rules (or Disjunctive Normal Forms) for each binary class representation and avoiding overlap, whether it is training sample or feature-space overlap, while maintaining interpretable explanations and perfect accuracy.

The Decision Tree model studied in this work consists in computing a minimum size decision tree, which would represent a 100% accurate classifier given a set of training samples. The model is based on encoding the problem as a CNF formula, which can be tackled with the efficient use of a SAT oracle.

Keywords: Explainable Artificial Intelligence, Interpretable Models, Prediction Interpretation and Justification, Decision Sets, SAT Solvers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Intelligence has been the subject of research for decades and its impact has been far-reaching since its birth on the Dartmouth Conference, in 1956. The proposal for the conference included this assertion: "every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it" [2]. From a simple game AI to self-driving cars, artificial intelligence is of great importance to technological advancement.

Machine Learning, a field of Artificial Intelligence, gives computers the ability to learn from data. Although very useful, most machine learning models nowadays are black boxes whose sole purpose is to predict, but we, as curious humans, need more than a (class) label. We need explicit knowledge in order to trust our models. With this in mind, comes Explainable Artificial Intelligence in which we understand why (and why not) some prediction was chosen, when the model will succeed (and when it will fail) and know when to trust the model itself (and why it erred) [3].

1.1 Motivation

Explainable Artificial Intelligence (XAI) is a promising and upcoming field of research [1, 4–6] with far-reaching expected impact. It also has an ongoing research program [3] and furthermore, the new European General Data Protection Regulations are enforcing automated generation of explanations [7]. There is also a number of meetings on computing machine learning models [8–10]. In a general sense, XAI aims to associate explanations with decisions made by autonomous agents. Even though there is an adequate amount of research on Artificial Intelligence [11, 12] there needs to be a shift in favor of Explainable Artificial Intelligence. A larger appeal needs to be made for the creation of XAI since the lack of transparency is a major drawback [13].

1.2 Objectives

The main purpose of this thesis is to summarize ongoing efforts in XAI by creating a knowledge base, assess models referenced and other available tools and, finally, create an Interpretable Model using logic-based methods with good performance, accuracy and a solid way of explaining decisions. This shall be achieved by:

- Developing a Knowledge Base of XAI through reference mapping;
- Understanding XAI models, assessing them and summarizing their pros and cons;
- Creating an Interpretable Model, with performance and accuracy comparable to existing classifiers, using logic-based reasoners;
- Comparing the proposed models against the state of the art tools in rule based learning, such as IDS, Orange, and Weka.

1.3 Contributions

With the completion of this thesis, we have achieved two main contributions:

- Bringing a more detailed logic approach to the world of Decision Sets, explaining different optimization problems with rigorous variables and constraints, with our models maintaining perfect accuracy when tested on the same training dataset, while minimizing the number of rules and providing approaches to minimize the number of literals as well, in order to bring an overall interpretable decision set;
- Learning an ideally minimum size Decision Tree with SAT techniques, with precise variables and constraints, helping us guess valid binary trees and verify whether or not they classify all the examples correctly.

1.4 Publications

During the period of my Master Thesis, I co-authored the following papers, which were accepted for publication in CORE A* conferences:

- *A SAT-Based Approach to Learn Explainable Decision Sets*, authored by A. Ignatiev, F. Pereira, N. Narodytska and J. Marques-Silva, which has been accepted for publication at IJCAR 2018 [14].
- *Learning Optimal Decision Trees with SAT*, authored by N. Narodytska, A. Ignatiev, F. Pereira, and J. Marques-Silva, which has been accepted for publication at IJCAI 2018 [15].

1.5 Document Organization

The document is organized as follows:

- Chapter 2 - Preliminaries: This chapter will revolve around the core concepts of this Thesis, like Boolean satisfiability, the Classification Problem, Complexity and the MaxSAT problem, minimal correction subsets and maximal satisfiable subsets.
- Chapter 3 - Related work: This chapter will focus on Interpretable Models and Frameworks based on Prediction Interpretation and Justification. There is also a section explaining the Smallest Decision Tree Problem [16], which is the inspiration for the logic-based model.
- Chapter 4 - Learning Decision Sets & Decision Trees with SAT: This chapter will explain in detail the work done throughout the duration of this thesis, from the design to the analysis of algorithms, as well as my contribution to the work done.
- Chapter 5 - Experimental Results: This chapter will show experimental results of the algorithm implemented in comparison with other tools.
- Chapter 6 - Discussion: This chapter will present a simple summary of the thesis, some final conclusions, address how the new legislation might affect the area of machine learning and what's to be expected of future work.

Chapter 2

Preliminaries

This section provides an overview of Boolean Satisfiability (SAT), the Classification Problem, Complexity, the MaxSAT Problem, Minimal Correction Subsets and Maximal Satisfiable Subsets.

2.1 Boolean Satisfiablity

We assume notation and definitions standard in the area of SAT [17]. Formulas are represented in Conjunctive Normal Form (CNF) and defined over a set of variables $X = \{x_1, ..., x_n\}$. A formula $\mathcal{F}$ is a conjunction of clauses, a clause is a disjunction of literals and a literal is a variable x_i or its respective complement $\neg x_i$. The Satisfiability Problem is the task of determining if a truth assignment (assignments of 0 or 1 to each variable) exists that makes a given propositional formula true.

2.2 Classification Problem

First of all, [R] is used to denote the set of natural numbers $\{1, ..., R\}$ and for a point f in some K-dimensional space, the r^{th} coordinates is given by $f[r]$. Following the notation used in earlier work [16], we consider a set of features $\mathcal{F} = \{f_1, ..., f_k\}$ which are considered to be binary, taking the value of either 0 or 1. When needed, the standard one hot encoding is used to handle non-binary categorical features, which turns labels into integers and integers into binary vectors. Since all features are binary, a literal on a feature f_r will be represented as f_r when the feature takes value 1 or $\neg f_r$ when it takes value 0. The feature space is represented by $\mathcal{U} \triangleq \prod_{r=1}^{K} \{f_r, \neg f_r\}$.

In order for a classifier to learn, it must begin from given training data $\mathcal{E} = \{e_1, ..., e_m\}$. Examples are associated with classes taken from a set of classes C but since these models focus mostly on binary classification ($C = \{c_o, c_1\}$), we will associate c_o with 0 and c_1 with 1. $\mathcal{E}$ is split into $\mathcal{E}^+$ and $\mathcal{E}^-$, denoting examples classified as positive and negative, respectively. Each example $e_q \in \mathcal{E}$ is represented as a 2-tuple (π_q, c_q), in which π_q denotes

the literals associated with the example and $c_q \in \{0, 1\}$ is the class to which the example belongs to. A literal l_r on a feature f_r, $l_r \in \{f_r, \neg f_r\}$, discriminates an example e_q if and only if $\pi_q = \neg l_r$, i.e the feature takes the value opposite to the value in the set of literals of the example.

Given the dataset presented at Table 2.1, two possible example decision sets are presented below, one with overlap [1] and another one without (the notion of overlap will be further discussed in 4.1.1), followed by an example decision tree and a brief explanation on how to present sample e_1:

Ex	Vacation(V)	Concert(C)	Meeting(M)	Expo(E)	Hike(H)
e_1	0	0	1	0	0
e_2	1	0	0	0	1
e_3	0	0	1	1	0
e_4	1	0	0	1	1
e_5	0	1	1	0	0
e_6	0	1	1	1	0
e_7	1	1	0	1	1

Table 2.1: A Classification Example

A decision set with some overlap would be:

If ¬ Meeting then Hike
If ¬ Vacation then ¬ Hike

A decision set with no overlap would be:

If Vacation then Hike
If ¬ Vacation then ¬ Hike

A decision tree for the dataset would be:

[1] Overlap between two rules assesses whether the set of points covered by two rules intersect (an example being the set of points $(\neg V, \neg C, \neg M, \neg E)$ for the decision set with overlap).

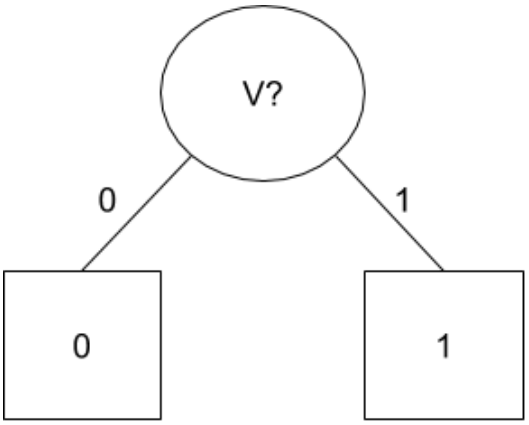

Figure 2.1: Decision Tree for Hike Dataset

The set of binary features is $\{f_1, f_2, f_3, f_4\}$, in which f_1 is the feature Vacation (V), f_2 the feature Concert (C), f_3 the feature Meeting (M) and f_4 the feature Expo (E). To this end, the example e_1 is represented by the tuple (π_1, c_1) with $\pi_1 = \{\neg f_1, \neg f_2, f_3, \neg f_4\}$ and $c_1 = 0$. Moreover, literals $\{f_1, f_2, \neg f_3, f_4\}$ discriminate e_1. Note that $\mathcal{U} = \{V, \neg V\} \times \{C, \neg C\} \times \{M, \neg M\} \times \{E, \neg E\}$.

The objective of this classification is to learn a function which matches the training data and generalizes suitably well on unseen test data. In this thesis, we seek representations of this function corresponding Decision Sets (DS) and Decision Trees (DT).

2.3 Complexity

Throughout this thesis, standard computational complexity definitions will be assumed. These include the well-known classes of NP-complete and NP-Hard problems. Additional classes of complexity, including different levels of the polynomial hierarchy, will also be assumed. The interested reader should check [18] for additional detail and information.

2.4 MaxSAT problem, Minimal Correction Subsets and Maximal Satisfiable Subsets

MaxSAT is an optimization version of SAT [19], and a MaxSAT problem consists of finding an assignment that satisfies the maximum number of clauses of a given unsatisfiable formula. The MaxSAT problem is well known to be NP-hard since SAT is polynomial time reducible to MaxSAT. A generalization called Partial MaxSAT exists, such that a subset of clauses that must be satisfied are named hard clauses, as for the remaining ones that may or may not be satisfied are named soft clauses. In a variant of the MaxSAT problem, named Weighted MaxSAT, each soft clause has an associated weight and the goal

becomes to maximize the sum of the weights of the satisfied soft clauses while satisfying all the hard clauses.

Based on earlier work [20], the largest Maximal Satisfiable Subset (MSS) represents a solution to the MaxSAT problem, which can also be represented by its complemented called the smallest Minimal Correction Subset (MCS).

Given a set of clauses $\mathcal{Z}$, which can be presented by z or z_i, with $i = \{1, ..., m\}$, where m = $|\mathcal{Z}|$, we have the following definitions:

- A MCS is defined as follows: $\mathcal{C} \subseteq \mathcal{Z}$, iff $\mathcal{Z} \setminus \mathcal{C}$ is satisfiable and $\forall z \in \mathcal{C}, \mathcal{Z} \setminus (\mathcal{C} \setminus \{z\})$ is unsatisfiable.
- A MSS is defined as follows: $\mathcal{S} \subseteq \mathcal{Z}$, iff $\mathcal{S}$ is satisfiable and $\forall z \in \mathcal{Z} \setminus \mathcal{S}, \mathcal{Z} \cup \{z\}$ is unsatisfiable.

In order to illustrate these concepts, we present the following example. The hard and soft clauses are, respectively: $H = \{(\neg x_1 \vee \neg x_2), (\neg x_1 \vee \neg x_3), (\neg x_4 \vee \neg x_5), (\neg x_4 \vee \neg x_6)\}$, $S = \{(x_1), (x_2), (x_3), (x_4), (x_5), (x_6)\}$.

Given the previous formula $\mathcal{Z}$, MaxSAT $\equiv 4$ with the MSS being the set of clauses $\{(x_2), (x_3), (x_5), (x_6)\}$ and MCS being the set of clauses $\{(x_1), (x_4)\}$ such that the union of both equals $\mathcal{Z}$. We pick this solution because it satisfies all the hard clauses and the largest number of soft clauses. Therefore, this solution has the lowest number of soft clauses left unsatisfied.

Chapter 3

Related Work

This chapter provides a first take at an annotated bibliography of papers that were deemed relevant to the thesis' main area of work.

In order to better represent the related work in XAI, a decision was made to categorize earlier work into two main areas, based on [21], those being Interpretable Models and Prediction Interpretation and Justification. Furthermore, the addition of a subsection was made in order to summarize the Smallest Decision Tree problem [16] that inspired a logic approach to machine learning.

The denotation Decision Lists and Rule Lists mean the same with these being ordered sets of "if-then" rules, furthermore, the denotation Decision Sets and Rule Sets also mean the same with these being unordered sets of "if-then" statements. More information is available in section 4.1.

3.1 Interpretable Models

There are a wide variety of interpretable models, whose performance and accuracy is high, but nowhere near as high as black box models. The trade-off is being able to provide explanations behind their predictions without the help of an additional framework. There is also a subsection to present other available tools, that are going to be assessed later in 5.5. Some examples are:

The work of Letham et al. [4] This paper aims to produce models that are highly accurate and interpretable by humans. These models are Bayesian Rule Lists, which consist of series of ordered if-then statements that discretize a high-dimensional, multivariate feature space into a series of simple, readily interpretable decision lists. Based on statistical rule learning, BRL produces a posterior distribution over permutations of if-then rules, starting from a large, pre-mined set of possible rules. A major source of BRL's practical feasibility is the fact that it uses these pre-mined rules, which reduce the model space to that of permutations of rules as opposed to all possible sets of splits thus reducing

the space complexity drastically. The following rule list (taken from [4] is based on the Titanic Dataset:

> **If** male **and** adult **then** *survival probability* 21% (19%-23%)
> **else if** 3rd class **then** *survival probability* 44% (38%-51%)
> **else if** 1st class **then** *survival probability* 96% (92%-99%)
> **else** *survival probability* 88% (82%-94%)

This model uses a statistical rule learning approach while ours uses an exact approach. Although BRL's performance is highly based on subsampling datasets, it doesn't reach a high accuracy on most of them, despite that, it is an innovative way to build decision sets.

The work of Lakkaraju et al. [1] This paper presents a framework for building predictive models that aim to be highly accurate and interpretable, but on our benchmarks (5.3) it failed to do so. The model created is a Decision Set which is simple, concise, interpretable and represents an unordered set of if-then rules. If a given feature is not found on the rules, then a default rule is used. Finding a model within a space designed for interpretability takes some time, so in order to work on big data, a pre-mined space of rules is needed.

A decision set $(\mathcal{S})$ is a set of rules of the form (π, c), where π is an itemset and c is a class label. An itemset π is a filter of data points, defined as a conjunction of one or more predicates of the form (attribute, operator, value) i.e $(x_1 > 5)$. The attribution of a class label c is as follows: If attribute values x satisfy exactly one itemset π_i, then the class label is the corresponding c_i. If x satisfies zero itemsets, then it is attributed to a default label. Finally, if x satisfies more than one itemset, it is assigned a class label c based on a tie-breaking function.

IDS tries to optimize interpretability and accuracy. For interpretability, IDS tries to:

- Lower the number of rules for easier reading, which is checked by $size(\mathcal{S})$;
- Have a decent number of predicates in a rule, which can be checked by $Length(r)$ for some rule $r = (\pi, c)$;
- Have a way to verify the set of data points which are satisfied in the Decision Set $\mathcal{S}$, defined on a per-rule basis by using $Cover(r)$ for a rule $r = (\pi, c)$, which is the set of data points in $\mathcal{S}$ with attribute values x that satisfy the itemset π;
- Lower the overlap, measured by $overlap(r, r^{'})$, for rules $r = (\pi, c)$ and $r^{'} = (\pi^{'}, c^{'})$, which checks the set of data points that satisfy both π and $\pi^{'}$. The measure is defined as: $overlap(r, r^{'}) = cover(r) \cap cover(r^{'})$.

In order to optimize accuracy, a decision set must effectively predict class labels. So IDS tries to measure per-rule accuracy with:

- $correct - cover(r)$, defined by: $correct - cover(r) = \{(x, y) \in cover(r) | y = c\}$ which is the set of data points in $\mathcal{S}$ that satisfy π and belong to class c;

- $incorrect - cover(r)$, defined by $incorrect - cover(r) = cover(r) \setminus correct - cover(r)$, which is the set of data points in $\mathcal{S}$ that satisfy π but do not belong to class c.

Optimization of said decision sets usually involves Smooth Local Search [22].

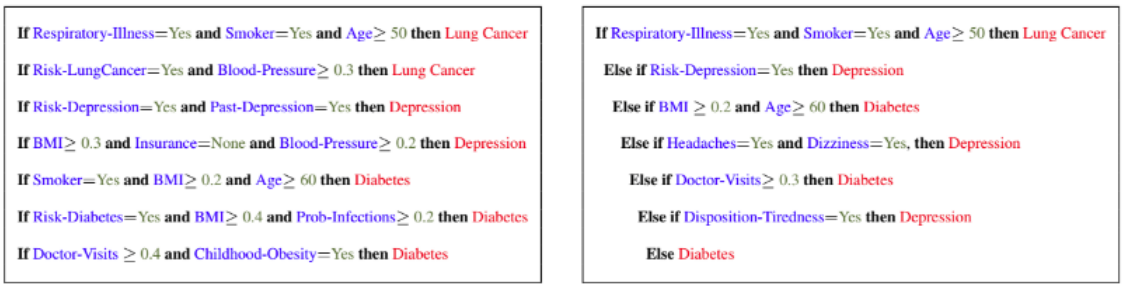

Figure 3.1: Interpretable Decision Set (on the left) and a Bayesian Decision List (on the right) on the same medical diagnosis dataset (taken from [1])

The framework IDS [1] was provided and even with a simple dataset such as Titanic with the .TAB extension (provided in the src), it took a very long time to build a decision set, averaging around 240 seconds. As soon as we put it to test with our datasets, this framework failed to provide solutions, which are explained in 5.3. Although this was unforeseen, IDS provided us with the definitions important to our work.

The work of Wang et al. [23] This paper presents the model Falling Rule Lists which consist of an ordered list of if-then rules where the order of the rules determines which example should be classified first by each rule and the estimated probability of success (or risk) decreases monotonically down the list. In certain situations, actions need to be taken and prioritized based on risk. The decision process is natural for a human decision-maker but not for machine learning models. Thus, falling rule list contains the most at-risk observation classified first. The model itself is quick to produce and serves a dual purpose: Ranking to form a predictive model and stratifying patients into decreasing risk sets. The model also aims to bring accuracy, interpretability, and computation. FRL builds the same way as Bayesian Rule Lists [4], with pre-mined itemsets.

	Conditions		Probability	Support
IF	IrregularShape AND Age ≥ 60	THEN malignancy risk is	85.22%	230
ELSE IF	SpiculatedMargin AND Age ≥ 45	THEN malignancy risk is	78.13%	64
ELSE IF	IllDefinedMargin AND Age ≥ 60	THEN malignancy risk is	69.23%	39
ELSE IF	IrregularShape	THEN malignancy risk is	63.40%	153
ELSE IF	LobularShape AND Density ≥ 2	THEN malignancy risk is	39.68%	63
ELSE IF	RoundShape AND Age ≥ 60	THEN malignancy risk is	26.09%	46
ELSE		THEN malignancy risk is	10.38%	366

Figure 3.2: Falling Rule List for mammographic mass dataset(taken from [23])

[1] https://github.com/lvhimabindu/interpretable_decision_sets/

The difference between Bayesian Rules Lists and Falling Rules Lists falls mostly on monotonicity and stratifying the most important cases first, the ones with a higher probability of risk (seen in 3.2).

The work of Angelino et al. [5] This paper implements a custom discrete optimization technique algorithm for building rule lists over a categorical feature space. The branch-and-bound algorithm named CORELS consists of a systematic enumeration of candidate solutions by means of feature space searching and provides a highly accurate rule list. Summarizing, this algorithm finds a transparent model that is optimal within a particular pre-determined class of models and produces a certificate of its optimality. The class of rule lists assembled are from pre-mined frequent itemsets and the objective is to search for a rule list that minimizes a regularized function. It also uses binarized datasets, just like in our approach.

CORELS (Certifiably Optimal Rules Lists) provides:

- An Optimal Solution;
- A Certificate of optimality;
- A Collection of near-optimal solutions and the distance between each one and the optimal one.

The following rule list, taken from [5], predicts two-year recidivism for the ProPublic dataset:

if (age = 23 - 25) **and** (priors = 2 - 3) **then predict** yes
else if (age = 18 - 20) **then predict** yes
else if (sex = male) **and** (age = 21 - 22) **then predict** yes
else if (priors $>$ 3) **then predict** yes
else predict no

The work of Kamath et al. [24] Inductive reasoning is a method of reasoning in which the premises are viewed as supplying strong evidence for the truth of the conclusion. This model creates a function $F : \{0,1\}^n \rightarrow \{0,1\}$ that maps input into an output, with this output being either 0 or 1, belonging respectively to the Off-set or On-set. The given function is represented as a sum of product terms (Disjunctive Normal Form). Explanations come from a logical form that is the disjunction of conjunction clauses. If the function F maps to 1, we can see which features have weight on the result based on the given input.

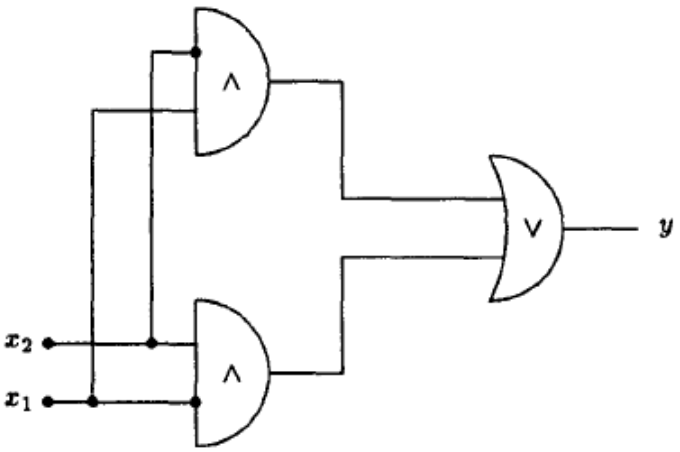

Figure 3.3: Circuit Representation of Boolean expression: $y = x_1\neg x_2 + \neg x_1 x_2$ (taken from [24])

Inspired by this earlier work, one of our models was created based on this approach, which is further studied in section 4.1.2.1.

The work of Li et al. [6] This paper implements a novel architecture for deep learning that explains its reasoning behind each prediction. Contains an autoencoder, which is a type of neural network used to learn efficient data codings in an unsupervised manner, and a prototype layer where each unit of that layer stores a weight vector that resembles an encoded training input.

The autoencoder has both an encoder (that allows comparisons within the latent space) and a decoder (that allows visualization of the learned prototypes). The decoder allows for a quick visualization of prototypes based on our dataset i.e if we train on the MNIST dataset, it will show the learned numbers.

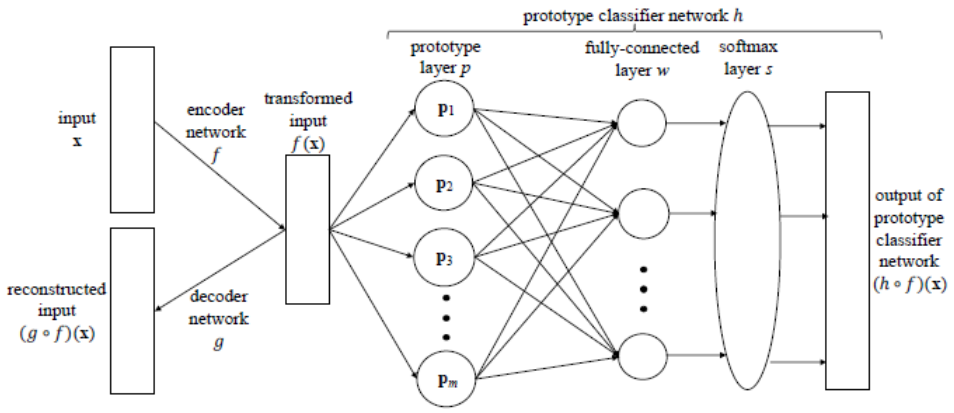

Figure 3.4: Network Architecture for Neural Network explanation (taken from [6])

The training objective has four terms: an accuracy term, a term that encourages ev-

ery prototype to be similar to at least one encoded input, a term that encourages every encoded input to be close to at least one prototype and a term that encourages faithful reconstruction by the autoencoder. Their definition of a prototype is something very close or identical to an observation in the training set and a set of prototypes is representative of the whole dataset.

The work of Lou et al. [25] Complex models for regression and classification have high accuracy but are unfortunately no longer interpretable by users. Generalized Additive Models (GAM) are complex functions turned into one-dimensional components so, in a way, these combine single-feature models called shape functions through a linear function that can be easily interpreted by users (by understanding the contribution of individual features). The model is fitted to the following form: $g(y) = f_1(x_1) + ... + f_n(x_n)$, where g is a link function, f_n a shape function and x_n a feature.

In order to create these models, we need to select a shape function for individual features (these being regression splines or ensembles of trees) and select a learning method for the model (learning square for regression splines and gradient boosting/backfitting for ensembles of trees).

The work of Clos et al. [26] Automatically classifying text documents is an active research challenge in document-oriented information systems. However, current approaches are biased towards building complex black box algorithms focused on producing high accuracy predictions at the cost of not being able to explain the rationale behind their decisions. This paper contributes with RELEXNET, an architecture that models lexicons as naive gated recurrent networks. Lexicon-based classifiers offer a white-box alternative to these approaches by using a trivially interpretable additive model at the cost of classification accuracy. This model is evaluated on stance detection and sentiment classification. Lexicons fill the need for XAI by offering a trivially interpretable additive model, where the probability of an instance belonging to a class is modeled as a weighted sum of the probabilities of each term of that class belonging to that class. Examining the terms of an instance and its weights allows us to understand a prediction.

3.1.1 Available tools

This subsection was created in order to reference three models that are going to be present in the Experimental Results section of this thesis. These models are going to be analysed in order to better understand how they compare with our models. All these models use a heuristic approach to rule learning.

Cohen's Model [27] JRIP is a class from the Weka collection that implements a propositional rule learner named Repeated Incremental Pruning to Produce Error Reduction

(RIPPER), which was proposed by William W. Cohen as an optimized version of IREP [27]. RIPPER is based on association rules with reduced error pruning (REP) which is a common technique used in decision tree algorithms. This model can use nominal and continuous features. JRIP follows this procedure:

Initialize Ruleset, RS = $\{\}$ and for each class, do:

1. Building Stage: Repeat step 1.a) and 1.b) until the description length (DL) of the ruleset and examples is greater than the smallest DL met so far, or there are no positive examples, or the error rate $\geq 50\%$
   (a) Grow Phase: Grow one rule by greedily adding antecedents to it until the rule is perfect.
   (b) Prune Phase: Incrementally prune each rule and allow the pruning of any final sequences of the antecedents.
2. Optimization Stage: After generating the initial ruleset $\{R_i\}$, generate and prune two variants of each rule R_i from randomized data using procedure 1.a) and 1.b). One variant is generated from an empty rule and the other is created by greedily adding antecedents to the original rule. The variant with the smallest DL is selected as the final representative of R_i in the rule set.
3. Delete rules from the ruleset that would increase the DL of the whole ruleset and add resultant ruleset to RS.

End do.

For more information on how the algorithm works, it is recommended to follow the footnote [2].

Cendrowska's Model [28] PRISM is a class from the Weka collection that implements a PRISM rule set for classification, which induces modular rules. This model only works with nominal attributes, can't deal with missing values and doesn't do any pruning. PRISM, although based on ID3, uses a different strategy to induce rules, avoiding problems associated with decision trees.

PRISM follows this procedure:

If the training set contains instances of more than one classification, then for each classification, c_n, do:

1. Calculate the probability of occurrence, $p(c_n|\alpha_x)$, of the classification c_n for each attribute-value pair α_x;
2. Select the α_x for which $p(c_n|\alpha_x)$ is a maximum and create a subset of the training set compromising all instances which contain the selected α_x;

[2] `http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/JRip.html`

3. Repeat steps 1 and 2 for this subset until it contains only instances of class c_n;
4. Remove all instances covered by this rule from the training set;
5. Repeat steps 1-4 until all instances of class c_n have been removed.

End do.

The difference between PRISM and ID3 is that PRISM focuses on finding only relevant values of attributes, while ID3 tries to find the attribute which is most relevant overall. ID3 also splits the training set into homogenous subsets (based on the most relevant attribute) while PRISM identifies subsets of a specific class.

Clark et al. Model [29] The Unordered List Learner CN2 algorithm induces a set of unordered rules. It is built on the Orange library, available for Python. It can work with numerical features but the binarization of these helps tremendously with accuracy. CN2 Unordered List Learner has a process of learning rules for each class individually.

The CN2 algorithm consists of two main procedures, a search algorithm performing a beam search for a good rule and a control algorithm for repeatedly executing the search heuristic. During the search procedure, CN2 must evaluate the rules searched, finding out which one is the best. CN2 has three different heuristics for this: Entropy, Laplace Accuracy and Weighted Relative Accuracy. By default, Laplace's rule of succession is used as a measure and is defined by: $LaplaceAccuracy = (n_c + 1)/(n_{tot} + k)$ where n_c is the number of examples in the predicted class c covered by the rule, n_{tot} is the total number of examples covered by the rule and k is the number of classes in the domain.

Now that the best rule is found, CN2 executes the search heuristic until we complete a rule. The control procedure for CN2's Unordered List Learner is shown in Figure 3.5.

```
procedure CN2unordered(allexamples, classes):
let ruleset = {}
for each class in classes:
   generate rules by CN2ForOneClass(allexamples,class)
   add rules to ruleset
return ruleset.

procedure CN2ForOneClass(examples,class):
let rules = {}
repeat
   call FindBestCondition(examples, class) to find bestcond
   if    bestcond is not null
   then add the rule 'if bestcond then predict class' to rules
      & remove from examples all exs in class covered by bestcond
until bestcond is null
return rules
```

Figure 3.5: Control Procedure for Unordered List Learner (taken from [24])

The main modification for the Unordered List Learner is for it to iterate the search for each class, removing only covered examples of that class when a rule has been found, unlike the Ordered List Learner which must maintain the negative examples because each rule must discriminate all negative examples. Also, for each class, the Laplace heuristic must be applied differently. With ordered rules, the predicted class c is taken simply as the one with most covered examples in it but with unordered rules, the predicted class is fixed to be the class selected by the revised control procedure.

3.2 Prediction Interpretation and Justification

Instead of making interpretable models, some authors tried a different approach that is based on building a framework to explain decisions of any classifier. Some good examples are:

The work of Ribeiro et al. [30] Machine Learning models remain mostly black boxes and nowadays understanding the reasons behind a prediction is needed in order to assess trust. LIME is a novel explanation technique that explains (presenting textual or visual artifacts) faithfully the prediction of any classifier by learning an interpretable model locally around the prediction.

The authors of this paper state that: *"If the users do not trust a model or prediction, they will not use it."*. The statement is true for decision making problems (examples being a medical diagnosis or terrorism detection) since we need to trust the prediction.

The desired characteristics for LIME are interpretability, local fidelity, model-agnostic, and a global perspective. In order to provide explanations, we need to assess which explanation might be the best. A model g contains a domain of $\{0,1\}^{d'}$ which is the absence/presence of interpretable components (these can be words or a patch of contiguous pixels). In classification, we have a function $f(x)$ which is the probability that x belongs to a certain class and $\pi_x(z)$ is a proximity measure between an instance z to x. Explanations by LIME are produced by the following formula:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

The objective is to minimize $\mathcal{L}$ (which is a measure of how unfaithful g is in approximating f in the locality defined by π_x) while maintaining Ω (which is the measure of complexity) low enough to be understandable by humans.

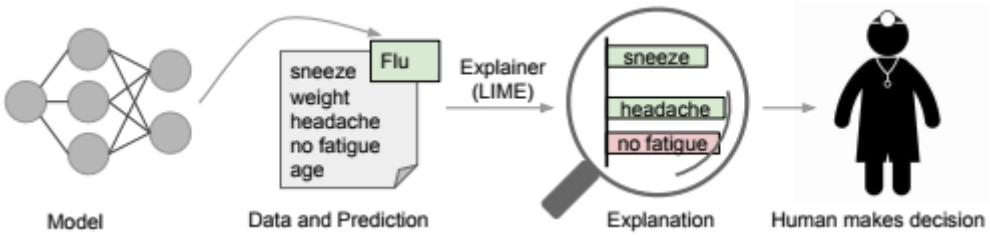

Figure 3.6: Explanation of Individual Predictions (taken from [30])

The work of Lakkaraju et al. [31] This paper proposes a model-agnostic framework, named Black Box Explanations through Transparent Approximations (BETA) that aims to explain any black box classifier while simultaneously optimizing fidelity of the model and the interpretability of the explanation.

The goal is to explain the behavior of any given black box classifier as a whole instead of just reasoning about its individual predictions. To this end, the framework constructs a small number of compact two-level decision sets, each of which captures the behavior of the given black box model in certain parts of the feature space. The framework also allows the user to define input, allowing him to explore the black box model.

The representation chosen is a two-level decision set (Figure 3.7) which can be seen as a set of multiple unordered decision sets, in which each is embedded with an outer if-then structure and the inner if-then rules present the decision logic employed by the black box model.

The framework has the following properties: Fidelity, Unambiguity, Interpretability, and Interactivity.

If Age <50 **and** Male =Yes:

If Past-Depression =Yes **and** Insomnia =No **and** Melancholy =No, **then** Healthy

If Past-Depression =Yes **and** Insomnia =Yes **and** Melancholy =Yes **and** Tiredness =Yes, **then** Depression

If Age ≥ 50 **and** Male =No:

If Family-Depression =Yes **and** Insomnia =No **and** Melancholy =Yes **and** Tiredness =Yes, **then** Depression

If Family-Depression =No **and** Insomnia =No **and** Melancholy =No **and** Tiredness =No, **then** Healthy

Default:

If Past-Depression =Yes **and** Tiredness =No **and** Exercise =No **and** Insomnia =Yes, **then** Depression

If Past-Depression =No **and** Weight-Gain =Yes **and** Tiredness =Yes **and** Melancholy =Yes, **then** Depression

If Family-Depression =Yes **and** Insomnia =Yes **and** Melancholy =Yes **and** Tiredness =Yes, **then** Depression

Figure 3.7: Explanations generated by BETA on Depression dataset (approximation on a Deep Neural Network) (taken from [31])

The work of Samek et al. [13] This paper appeals to bring more interpretability in Artificial Intelligence since the lack of it is a major drawback and tries to provide explanations in Deep Learning Models, implementing two approaches for it:

The first approach is called Sensitivity Analysis which explains a prediction based on the model's locally evaluated gradient, computing the sensibility of the prediction regarding the input. The quantification for the importance of each input variable i is defined as $R_i = ||\frac{\partial}{\partial x_i} f(x)||$.

The second approach is called Layer-Wise Relevance Propagation which explains the classifiers decisions by decomposing the decision in terms of input variables. Mathematically, it redistributes the prediction f(x) backwards using local redistribution rules until it assigns a relevance score R_i to each input variable.

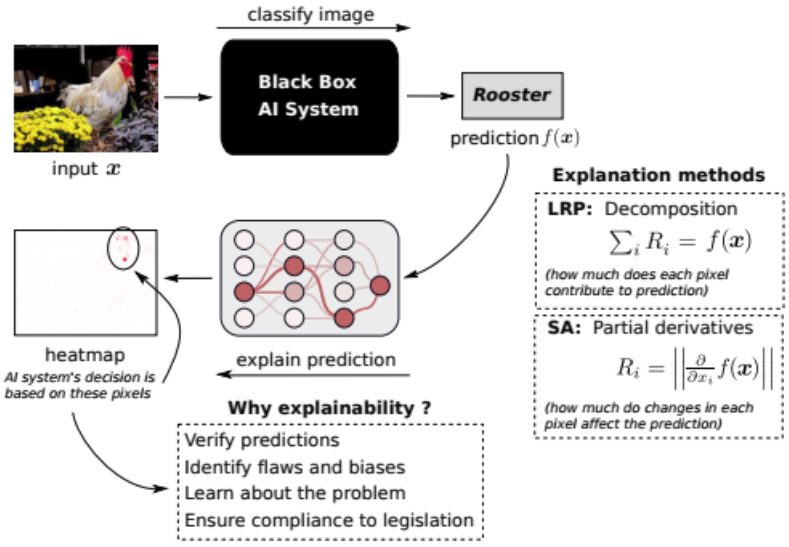

Figure 3.8: Explaining predictions of an AI System (taken from [13])

3.3 Smallest Decision Tree Problem

The Smallest Decision Tree Problem is taken from the paper *Minimizing Decision Tree Size as Combinatorial Optimisation* [16]. The objective is to minimize the decision tree size by regarding the learning task as a combinatorial optimization problem in which the objective is to minimize the number of nodes in the tree.

The following table (3.1) presents the problem of finding the smallest decision tree that is consistent with a set of training examples:

Key Concepts	Description
$\mathcal{E} = \{e_1, ..., e_m\}$ is a set of examples, Let $\mathcal{E}^+$ and $\mathcal{E}^-$ be partitions of $\mathcal{E}$	$\mathcal{E}^+$ and $\mathcal{E}^-$ represent the positive and negative examples of $\mathcal{E}$, respectively
$\mathcal{F} = \{f_1 ..., f_k\}$ is a set of features	$e[f]$ is the evaluation (0 or 1) of feature $f \in \mathcal{F}$ in the example $e \in \mathcal{E}$
$T = (X, U, r)$ is a binary tree rooted by $r \in X$	$L \subseteq X$ is the set of leaves of T Internal nodes $x \in X \setminus L$ are labeled by $f(x) \in \mathcal{F}$
$(x, y) \in U$ is an edge labeled with Boolean $g(x, y)$	$g(x, y) = 0$ if y is the left child of x $g(x, y) = 1$ if y is the right child of x
$p(l)$ is a path in T	for $l \in L$ denotes the path in T from the root r to leaf l
$\forall e \in \mathcal{E}$ associate the unique leaf $l(e) \in L$	every edge (x, y) in $(p(l(e))$ has $e[f(x)] = g(x, y)$

Table 3.1: Formulation of the problem of finding the Smallest Decision Tree

With these concepts taken from [16], we need to find a way to build a decision tree that evaluates $\mathcal{E}$ with the fewest nodes possible or alternatively lowering the longest branch.

3.3.1 Sat-Based Encoding

In order for this SAT model to find a smaller decision tree, it requires a large number of clauses to represent the problem. Given a fixed tree $T = (X, U, r)$ and a set of examples $\mathcal{E}$, a formula is presented that is satisfiable if and only if there is a decision tree based on T that classifies $\mathcal{E}$.

Intuition. Given a set of features $\mathcal{F} = \{a, b, q, r\}$, suppose there are two examples $e_i \in \mathcal{E}^+$ and $e_j \in \mathcal{E}^-$ that have an equal value on the set of features $eq(e_i, e_j) = \{a, b\}$, $(e_i[a] = e_j[a] = 0, e_i[b] = e_j[b] = 1)$, and that differ on the set of values $\mathcal{F} \setminus eq(e_i, e_j) = \{q, r\}$. Even though they have the same values, the encoding must be done in a way that e_i and e_j are not associated with the same leaf l.

The only way to do this is if and only if there exists an edge $(x, y) \in p(l)$ such that:

$$f(x) \in \mathcal{F} \setminus eq(e_i, e_j) \vee (f(x) \in eq(e_i, e_j) \wedge g(x, y) \neq e_i[f(x)])$$

The first part of the formula ensures that if $l(e_i)$ and $l(e_j)$ have x as a common ancestor, they both appear in one of the subtrees rooted in x. The second part ensures that none of $l(e_i)$ and $l(e_j)$ are equal to l since they will both branch on the opposite child of x.

Encoding. For every node $x \in X \setminus L$ and for every feature $f \in \mathcal{F}$, a literal $t_x f$ is introduced, whose value 1 means the node x is labeled with the feature f. For each pair $e_i \in \mathcal{E}^+, e_j \in \mathcal{E}^-$ and for each $l \in L$, a clause is built that forbids e_i and e_j to be classified at leaf l.

Suppose, for the example above, that there is a path $p(l) = (x_1, x_2, l)$ in the tree such as x_2 is the left child of x_1 and l is the right child of x_2, the following clause needs to be added: $t_{x_1q} \vee t_{x_1r} \vee t_{x_2q} \vee t_{x_2r} \vee t_{x_1b} \vee t_{x_2a}$.

The variable t_{x1q} means that x_1 is labeled with a feature that discriminates e_i and e_j because $q \in \mathcal{F} \setminus eq(e_i, e_j)$. The variable t_{x1b} means the feature labeling x_1 will classify both e_i and e_j in the branch that does not lead to l because p(l) uses the left child of x_1 whereas $e_i[b] = e_j[b] = 1$. Let us define the following equation as Equation 1 and the clauses built are:

$$\begin{gathered}(\bigvee_{(x,y)\in p(l), f\in e_q(e_i,e_j)|g(x,y)\neq e_i[f]} t_{xf}) \\ \vee \quad (\bigvee_{(x,y)\in p(l), f\in\mathcal{F}\setminus e_q(e_i,e_j)} t_{xf}) \\ \forall (e_i, e_j) \in \mathcal{E}^+ \times \mathcal{E}^-, \forall l \in L.\end{gathered} \tag{1}$$

Finally, the following clause, named Equation 2, ensures that each node is labeled with at most one feature:

$$(\neg t_{xf} \vee \neg t_{xf'}), \forall x \in X \setminus L, \forall f, f^{'} \in \mathcal{F}. \tag{2}$$

A solution to the formula presented above characterizes a Decision Tree. Let M be such a solution, $x \in X \setminus L$ will be labeled with $f \in \mathcal{F}$ if and only if $M[t_{xf}] = 1$. Redundant clauses are added specifying that two nodes on the same path should not take the same features (this speeds up the resolution process). Equation 3 is as follows:

$$\bigwedge_{(x,y)\in p(l),(x',y')\in p(l),x\neq x'} (\neg t_{xf} \vee \neg t_{x'f}), \forall l \in L, \forall f \in \mathcal{F}. \tag{3}$$

3.3.2 Space Complexity

Given $N =\mid X \mid$, $K =\mid \mathcal{F} \mid$, $M =\mid \mathcal{E} \mid$, the number of literals is $\mathcal{O}(NK)$. The space (or encoding) complexity (or size) of this problem can be described by the following table:

Equation Type	Clauses Built	Length of Clauses	Space Complexity
(1)	$M^2 \times N/2$	$\mathcal{O}(NK)$	$\mathcal{O}(KN^2M^2)$
(2)	$N/2 \times K^2$	$\mathcal{O}(1)$	$\mathcal{O}(NK^2)$
(3)	$N/2 \times K$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3K)$

Table 3.2: Space complexity of the SAT-Based Encoding of Smallest Decision Tree problem

The full space-complexity of this problem is $\mathcal{O}(K \times N^2 \times M^2 + N \times K^2 + K \times N^3)$.

The main problem lies in the fact that in order for the algorithm to work, it needs a given fixed tree T. The algorithm then attempts to run a perfect classification on the training data but one drawback is its large space complexity. From Equation 2 and Equation 3, further study and examination will bring to a conclusion that these are hidden AtMost1 constraints. There are simpler ways to encode this, thus reducing space complexity effectively. Since Bessiere et al. didn't offer any results for the SAT-Based Encoding approach, due to its Conjuctive Normal Forms size being huge (visible in 3.9), any results based on a logic approach would be helpful.

Benchmark	Weather	Mouse	Cancer	Car	Income	Chess	Hand w.	Magic	Shuttle	Yeast
#examples	14	70	569	1728	30162	28056	20000	19020	43500	1484
#features	10	45	3318	21	494	40	205	1887	506	175
CNF size (depth 4)	27K	3.5M	92G	842M	354G*	180G	248G	967G*	118G*	13G

Figure 3.9: Benchmarks of various datasets and corresponding SAT Formulae size (taken from [16])

Chapter 4

Learning Decision Sets & Decision Trees with SAT

The objective of this chapter is to show the analysis and implementation of an algorithm that is interpretable and accurate, based on logic reasoners. Although two algorithms were produced, my focus will be on the Decision Set model. The papers *A SAT-Based Approach to Learn Explainable Decision Sets* [14] and *Learning Optimal Decision Trees with SAT* [15] were accepted for publication in CORE A* Conferences, respectively, IJCAR and IJCAI. Given that I am a co-author of both, there will be some similarities between this chapter and previously mentioned papers.

4.1 A SAT-Based Approach to Learn Explainable Decision Sets

Machine Learning has made remarkable progress and one approach often used to provide explanations is the creation of Decision Lists and/or Decision Sets, which are sets of ”if-then” statements. Both can be represented as formulas in a clausal form. Decision Lists impose an order of the rules connected by ”else” statements while Decision Sets do not. From an interpretable perspective, decision sets seem to be the simpler choice but unfortunately, decision sets can exhibit rule overlap. Restricted forms of rule learning are also known to be NP-Hard [32].

4.1.1 Learning Explainable Decision Sets

This section introduces a generalization of the definitions proposed in earlier work [1] and the generation of succint explanations. For more information on this earlier work, it is advisable to check 3.1.

4.1.1.1 Definitions

Earlier work paved the road to Decision Sets, so here are the generalization of the definitions (with some changes, based on our models):

Definition 1: (Itemset). Given $\mathcal{F}$, an itemset π is an element of $\mathcal{I} \triangleq \prod_{r=1}^{K}\{f_r, \neg f_r, \mathfrak{u}\}$, where $\mathfrak{u}$ represents a "don't care" value.

The itemset of earlier work [1] is defined as a conjunction of one or more predicates of the form (attribute, operator, value). Since our case is a binary approach, the form will be a Boolean literal i.e "Vacation" or " $\neg$ Vacation".

Definition 2: (Clashing itemsets). Given two itemsets, $\pi_1, \pi_2 \in \mathcal{I}$, the two itemsets clash, written $\pi_1 \cap \pi_2 = \emptyset$, if and only if there exists a coordinate r such that $\pi_1[r] = f_r \wedge \pi_2[r] = \neg f_r$ or $\pi_1[r] = \neg f_r \wedge \pi_2[r] = f_r$.

Definition 3: (Rule). A rule is a 2-tuple (π, c), where $\pi \in \mathcal{I}$ is an itemset and $c \in C$ is a class. A rule can be interpreted as follows:

IF the specified literals in π are true, THEN pick class c.

Given Table 2.1, the decision set with no overlap can be represented as the following rules: Rule1: $((V, \mathfrak{u}, \mathfrak{u}, \mathfrak{u}), c_1) \wedge$ Rule2: $((\neg V, \mathfrak{u}, \mathfrak{u}, \mathfrak{u}), c_o) \wedge$ Default rule $\mathcal{D} : (\emptyset, c_0)$.

Definition 4: (Decision Sets) Given a set of binary features $\mathcal{F}$, defining a feature space $\mathcal{U}$, and a set of classes $\mathcal{C}$, a decision set $\mathcal{S}$ is a finite set of rules.

Given a Decision Set $\mathcal{S}$, there may exist points in the feature space that are not covered by $\mathcal{S}$. A solution is the addition of a default rule which is explained in the following definition.

Definition 5: (Default Rule $\mathcal{D}$) A rule of the form $\mathcal{D} \triangleq (\emptyset, c)$ denotes the default rule of a decision set S. This rule applies whenever the previous rules are not satisfied (give a value of 0) for every point on the feature space.

Given 2.1 and the decision set with some overlap, one (necessary) default rule would be $(\emptyset, 0)$. For a feature space point (V, C, M, E), we can conclude the class is 0 due to the default rule.

Definition 6: ($\mathcal{X}$-cover) Given $\mathcal{X} \subseteq \mathcal{U}$ and an itemset π, the $\mathcal{X}$-cover of the itemset is the set of feature space points in $\mathcal{X}$ with a non-empty intersection with the itemset. Earlier work [1] considers a less general definition of cover, where $\mathcal{X}$ corresponds to the training

data $\mathcal{E}$. Overlap between two rules assesses whether the set of points covered by the two rules intersect. Earlier work has focused solely on overlap with respect to the training data.

Definition 7: ($\mathcal{X}$-overlap) Two rules $r_1 = (\pi_1, c_1)$ and $r_2 = (\pi_2, c_2)$ overlap in $\mathcal{X} \subseteq \mathcal{U}$ if and only if:

$$\exists f \in \mathcal{X}. f \cap \pi_1 \neq \emptyset \wedge f \cap \pi_2 \neq \emptyset \tag{4}$$

Definition $\pi_1 \cap \pi_2 \neq 0$ would not enable restricting overlap to specific subsets of $\mathcal{U}$. The definition of overlap considered in earlier work [1] corresponds to $\mathcal{E}$-overlap.

The definition above can be qualified with $\oplus$ or $\ominus$, depending on the following condition for each, respectively:

- Overlap where the classification agrees (all rules that are not false predict the same class);
- Overlap where the classification disagrees (there exist rules that are not false that do not predict the same class).

This formulation allows for better quality decision sets since we can search for feature space points not used in the samples. Given Table 2.1 again, if we pick the decision set with some overlap, we notice there is no $\mathcal{E}$-overlap. But, for the point $(\neg V, \neg C, \neg M, \neg E) \in \mathcal{U}$ we have feature space overlap ($\mathcal{U}^{\ominus}$-overlap).

4.1.1.2 Generating Succinct Explanations

For a rule (π, c), its explanation is the conjunction of literals in π. So, if for any point in the feature space there exists no $\ominus$-overlap, we can pick a rule consistent with that point for the explanation. This explanation is referred to as offline (or explicit). If $\ominus$-overlap exists, we can pick one of the rules for which the itemset takes value 1 and list the itemset as an explanation. When a feature space point is not covered by any rule in the decision set, we resort to the default rule $\mathcal{D} = (\emptyset, 0)$ which has no immediate explanation. Although it is still able to provide explanations, we need to find the literals that falsify the itemset in the feature space point. So, the explanation is picked by the falsified literals from each itemset that is not consistent with the class associated with the default rule. These explanations are online (or implicit).

4.1.2 Learning Decision Sets with SAT

This section details a number of SAT models to learn decision sets. Beforehand, it is important to mention that the abbreviation MinDS represents Miner of Decision Sets. We will associate a Boolean function E^0 with $\mathcal{E}^-$, which takes value 1 for each point in

feature space associated with $\mathcal{E}^-$ i.e each combination of binary features that represents an example in $\mathcal{E}^-$ is a minterm of E^0. The same applies to E^1.

Our working hypothesis is that $E^0 \wedge E^1 \models \bot$. Our approach to the minimum decision set problem is a general formalization, computing two sets of terms F^0 and F^1, which equal to two Disjunctive Normal Forms:

Definition 8: [MinDSet,MinDS$_0$] Let $\{\mathcal{E}^-, \mathcal{E}^+\}$ be a tuple of examples associated with two distinct classes, c_0 and c_1 and each represented by E^0, E^1, respectively. MinDS$_0$ is the problem of finding the smallest DNF representation of Boolean function F^0 and F^1, measured in the number of terms (rules), such that: (i) $E^0 \models F^0$ (ii) $E^1 \models F^1$ (iii) $F^1 \Leftrightarrow F^0 \models \bot$.

$\mathcal{U}^{\ominus}$-overlap is prevented if any feature space point that is true for E^0 is also true for F^0 with the same conditions applying to E^1 and F^1. Condition (iii) also ensures that a decision set is computed covering the complete feature space $\mathcal{U}$. The cost of DNF representation could be measured by the number of literals but our approach took into account the cost in terms (number of rules).

Lemma 1. For any decision set respecting Definition 8, it holds that i) $F^0 \wedge E^1 \models \bot$ and ii) $F^1 \wedge E^0 \models \bot$.

Proposition 1: The decision version of MinDS$_0$ is in Σ_2^p.

Proof. (Sketch - [14]) Given some size threshold T, simply guess the terms of two DNFs, F^0 and F^1 using no more than T terms and then check that, for every assignment, the values of F^0 and F^1 differ.

Conjecture 1: MinDS$_0$ is hard for Σ_2^p.

The proof (or disproof) of this conjecture is beyond the scope of this thesis. With the previous conjecture in mind, we can picture the following optimization problems which result from relaxing the constraint $F^1 \Leftrightarrow F^0 \models \bot$ of MinDS$_0$ thus achieving hardness for NP:

1. MinDS$_4$: Minimize F^0, given $F^1 \equiv E^1$ constant, and such that (i) $E^0 \models F^0$; and (ii) $F^0 \wedge E^1 \models \bot$.
2. MinDS$_3$: Same as above, but for F^1 given $F^0 \equiv E^0$ constant.
3. MinDS$_2$: Minimize both F^0 and F^1, such that (i) $E^0 \models F^0$; (ii) $E^1 \models F^1$; (iii) $F^0 \wedge E^1 \models \bot$; and (iv) $F^1 \wedge E^0 \models \bot$.
4. MinDS$_1$: Minimize F^0 and F^1, such that (i) $E^0 \models F^0$; (ii) $E^1 \models F^1$; and (iii) $F^0 \wedge F^1 \models \bot$.

All these problems are weakened versions of MinDS_0, the main difference being the constraints on functions associated with E^0 and E^1.

Proposition 2: The decision versions of these optimization problems are complete for NP.

Proof. (Sketch - [14]) It is possible to reduce MinDS_3 and MinDS_4 to MinDS_1 or MinDS_2. Moreover, the decision versions of MinDS_1 and MinDS_2 are in NP if we reduce these problems to SAT.

Given Table 2.1 and its respective decision sets, the decision set with some overlap respects MinDS_4, MinDS_3 and MinDS_2, whereas the decision set with none respects MinDS_1 and MinDS_0.

Further notation will use N for the number of terms (rules), M for the number of samples in the training dataset and K for the number of literals.

4.1.2.1 SAT Models for MinDS_3 and MinDS_4

This section details SAT Models for solving MinDS_3 but with some minor modifications, similar models can be devised for MinDS_4. The purpose of MinDS_3 is to find a minimum size representation of F^1, subject to a non-$\mathcal{U}^{\ominus}$-overlap constraint with respect to E^0. This model considers a grid of N by K entries, each row of K entries denoting the representation of the condition of a rule or, alternatively, a term in the DNF representation of F^1, for a total of N terms. Throughout this section, it holds that $1 \leq j \leq N$ and $1 \leq r \leq K$, with q associated with some example e_q from $\mathcal{E}$.

Model based on existing SAT model: This model, based on [24], assumes the representation of a Boolean function in terms of K-dimensional points describing the functions ON-set and OFF-set, respectively E^1, E^0. Variables used for this representation are:

1. $p_{jr} = 1$, if and only if x_i is not included in rule j, 0 otherwise;
2. $p^{'}_{jr} = 1$, if and only if $\neg x_i$ is not included in the rule j, 0 otherwise;
3. sl^q_{jr} is a variable that replaces either with $p^{'}_{jr}$ if the feature f_r occurs positively in $e_q \in \mathcal{E}^+$, or with p_{jr} if feature f_r occurs negatively in $e_q \in \mathcal{E}^-$;
4. $cr_{jq} = 1$, if and only if rule j covers $e_q \in \mathcal{E}^+$.

The constraints added in order to create valid rules and decision sets are:

1. One of p_{jr} and $p^{'}_{jr}$ must be true:

$$(p_{jr} \vee p'_{jr}), j \in [N] \wedge r \in [K] \tag{5}$$

2. Any negative example $e_q \in \mathcal{E}^-$, with a set of positive features P_q and a set of negative features N_q, must be discriminated by any term:

$$(\bigvee_{r \in P_q} \neg p'_{jr} \vee \bigvee_{r \in N_q} \neg p_{jr}), j \in [N] \wedge e_q \in \mathcal{E}^- \tag{6}$$

3. Each positive example must be covered:
   - Constraint for a term not covering a positive example:

$$(sl^q_{jr} \vee \neg cr_{jq}), j \in [N] \wedge r \in [K] \wedge e_q \in \mathcal{E}^+ \tag{7}$$

   - Each positive example must be covered by some term:

$$\left(\bigvee_{j=1}^{N} cr_{jq}\right), e_q \in \mathcal{E}^+ \tag{8}$$

This model uses $\mathcal{O}(N \times M \times K)$ clauses and literals.

Example 1. *Given Table 4.1, the following decision set would be produced:*

Football	Work	HaveBeer
1	0	1
1	1	1
0	1	0

Table 4.1: HaveBeer Dataset

'Football: 1'⇒ HaveBeer: 1
not 'Football: 1', ' Work: 1' ⇒ HaveBeer: 0

In order to get the rule 'Football: 1'⇒ HaveBeer: 1 for the positive class, we need the following:

Variables: $\{p_{1,1}, p'_{1,1}, p_{1,2}, p'_{1,2}, cr_{1,1}, cr_{1,2}\}$

Constraints:

- $(5).(p_{1,1} \vee p'_{1,1}), (p_{1,2} \vee p'_{1,2})$

- $(6).(\neg p^{'}_{1,2} \vee \neg p_{1,1})$
- $(7).(p^{'}_{1,1} \vee \neg cr_{1,1}), (p_{1,2} \vee \neg cr_{1,1}), (p^{'}_{1,1} \vee cr_{1,2}), (p^{'}_{1,2} \vee \neg cr_{1,2})$
- $(8).(cr_{1,1}), (cr_{1,2})$

The solution is: $[\neg p_{1,1}, p^{'}_{1,1}, p_{1,2}, p^{'}_{1,2}, cr_{1,1}, cr_{1,2}]$.

With the purpose of assessing the efficiency of SAT solvers, we developed a new, **Alternative Model** with different semantics for some variables and additional clauses, to elicit propagation.

The variables used are:

1. s_{jr}: whether for rule j, a literal in feature r is to be skipped;
2. l_{jr}: literal on feature r for rule j, in case the feature is not to be skipped;
3. d^0_{jr}: whether feature r of rule j discriminates value 0;
4. d^1_{jr}: whether feature r of rule j discriminates value 1;
5. cr_{jq}: whether rule j covers $e_q \in \mathcal{E}^+$.

The constraints for encoding MinDS$_3$ are as follows:

1. Each term must have literals:

$$\left(\bigvee_{r=1}^{K} \neg s_{jr} \right), j \in [N] \tag{9}$$

2. One must account for which literals are discriminated by which rules:

$$\begin{aligned} d^0_{jr} &\leftrightarrow \neg s_{jr} \wedge l_{jr}, j \in [N] \wedge r \in [K] \\ d^1_{jr} &\leftrightarrow \neg s_{jr} \wedge \neg l_{jr}, j \in [N] \wedge r \in [K] \end{aligned} \tag{10}$$

3. In addition, one must be able to discriminate all negative examples in each term. Let $e_q \in \mathcal{E}^-$ be a negative example and $\sigma(r, q)$ denote the value of feature f_r for e_q.

$$\left(\bigvee_{r=1}^{K} d^{\sigma(r,q)}_{jr} \right), j \in [N] \wedge e_q \in \mathcal{E}^- \tag{11}$$

4. We must also ensure that each positive example is covered by some rule associated with its class:
   - First, we define whether a rule covers some specific positive example:

$$cr_{jq} \leftrightarrow \left(\bigwedge_{r=1}^{K} \neg d^{\sigma(r,q)}_{jr} \right), j \in [N] \wedge e_q \in \mathcal{E}^+ \tag{12}$$

- Second, each $e_q \in \mathcal{E}^+$ must be covered by some term. Same equation as (8).

$$\left(\bigvee_{j=1}^{N} cr_{jq}\right), e_q \in \mathcal{E}^+$$

This encoding also uses $\mathcal{O}(N \times M \times K)$ clauses and literals.

Example 2. *Given Table 4.1, the following decision set would be produced:*

'Football: 1'⇒ HaveBeer: 1
not 'Football: 1'⇒ HaveBeer: 0

In order to get the rule 'Football: 1'⇒ HaveBeer: 1 for the positive class, we need the following:

Variables: $\{s_{1,1}, s_{1,2}, l_{1,1}, l_{1,2}, d^0_{1,1}, d^0_{1,2}, d^1_{1,1}, d^1_{1,2}, cr_{1,1}, cr_{1,2}\}$

Constraints:

- $(9).(\neg s_{1,1} \vee \neg s_{1,2})$
- $(10).(d^0_{1,1} \leftrightarrow \neg s_{1,1} \wedge l_{1,1}), (d^0_{1,2} \leftrightarrow \neg s_{1,2} \wedge l_{1,2})$
  $(d^1_{1,1} \leftrightarrow \neg s_{1,1} \wedge \neg l_{1,1}), (d^1_{1,2} \leftrightarrow \neg s_{1,2} \wedge \neg l_{1,2})$
- $(11).(d^0_{1,1} \vee d^1_{1,2})$
- $(12).(cr_{1,1} \leftrightarrow \neg d^1_{1,1} \wedge \neg d^0_{1,2}), (cr_{1,2} \leftrightarrow \neg d^1_{1,1} \wedge \neg d^1_{1,2})$
- $(8).(cr_{1,1}), (cr_{1,2})$

The solution is: $[\neg s_{1,1}, s_{1,2}, l_{1,1}, \neg l_{1,2}, d^0_{1,1}, \neg d^0_{1,2}, \neg d^1_{1,1}, \neg d^1_{1,2}, cr_{1,1}, cr_{1,2}]$.

4.1.2.2 SAT Models for MinDS$_2$ and MinDS$_1$

The models analyzed in the previous section learn one function for one class, i.e F^1 for c_1. For the other class, c_0, only the original minterms are available and a default rule that may opt to pick this class for other points of feature space not covered by F^1.

Case for MinDS$_2$: It is simple to generalize MinDS$_3$/MinDS$_4$ to the case of MinDS$_2$. MinDS$_2$ is the same as MinDS$_3$ or MinDS$_4$ but it considers two function representations F^0 and F^1 instead of just F^1 or F^0, respectively. So, essentially, in order to generalize into MinDS$_2$, we just need to replicate the constraints for discriminating classes and for

covering classes for the target classes c_0 and c_1.

Case for MinDS$_1$: A grid of N by K entries is considered, each row of K entries denotes the organization of a rule. The set of variables to use are the same as for MinDS$_3$, with the addition of c_j representing a class variable, which is 0 if the class of rule j is false and 1 otherwise. The constraints for MinDS$_1$ are:

1. Every term must be used (same equation as (9)):

$$\left(\bigvee_{r=1}^{K} \neg s_{jr}\right), j \in [N]$$

2. We must also be able to account for which literals are discriminated by which rules (same equation as (10)):

$$\begin{aligned} d_{jr}^{0} &\leftrightarrow \neg s_{jr} \wedge l_{jr}, j \in [N] \wedge r \in [K] \\ d_{jr}^{1} &\leftrightarrow \neg s_{jr} \wedge \neg l_{jr}, j \in [N] \wedge r \in [K] \end{aligned}$$

3. In addition, we must be able to discriminate positive examples in rules of negative class and vice-versa. Let $e_q \in \mathcal{E}^+$ be a positive example and $\sigma(r, q)$ be defined as above. Then:

$$\begin{aligned} \neg c_j &\rightarrow \left(\bigvee_{r=1}^{K} d_{jr}^{\sigma(r,q)}\right), j \in [N] \wedge e_q \in \mathcal{E}^+ \\ c_j &\rightarrow \left(\bigvee_{r=1}^{K} d_{jr}^{\sigma(r,q)}\right), j \in [N] \wedge e_q \in \mathcal{E}^- \end{aligned} \tag{13}$$

4. We must also ensure that each example is covered by some rule, associated with its class.

   - First, the constraint for a rule to cover some sample is:

$$\begin{aligned} cr_{jq} &\leftrightarrow \left(\neg c_j \wedge \bigwedge_{r=1}^{K} \neg d_{j,r}^{\sigma(r,q)}\right), j \in [N] \wedge e_q \in \mathcal{E}^- \\ cr_{jq} &\leftrightarrow \left(c_j \wedge \bigwedge_{r=1}^{K} \neg d_{j,r}^{\sigma(r,q)}\right), j \in [N] \wedge e_q \in \mathcal{E}^+ \end{aligned} \tag{14}$$

   - Second, all examples, no matter the class must be covered, so we generalize (8) to get:

$$\left(\bigvee_{j=1}^{N} cr_{jq}\right), e_q \in \mathcal{E} \tag{15}$$

   This means that every element is now covered.

5. Finally, two terms associated with different classes must not exhibit $U^{\ominus}$-overlap.

$$\neg(c_i \leftrightarrow c_j) \rightarrow (\bigvee_{r=1}^{K} \neg s_{ir} \wedge \neg s_{jr} \wedge \neg(l_{ir} \leftrightarrow l_{jr}), i, j \in [N] \wedge i < j \tag{16}$$

4.1.3 Symmetry Breaking for SAT Models

These propositioned models essentially capture unordered sets of terms. This lack of order reveals a symmetry and if the number of terms is large, this can impact performance significantly. A standard technique used to eliminate such problems is to impose an order in the representation, so we decide to sort terms, such that each feature is inverse to the weight of the feature in the binary representation of the number associated with the term. Unspecified features have the largest weight. Clearly, imposing an order on the term does not affect correctness but it does affect complexity.

We describe the symmetry breaking predicates for the alternative model proposed in section 4.1.2.1. For the other models, a similar approach is used. The additional variables added are as follows:

- $eq_{j,r} = 1$, if and only if term j equals term $j-1$ until feature r;
- $gt_{j,r} = 1$, if and only if term j is greater than term $j-1$ by feature r.

For the constraints below, we have $j \in [N]$ and $r \in [K]$. The constraints for eq_{jr} (with $eq_{j,0}$ = 1) are as follows:

$$eq_{j,r} \leftrightarrow eq_{j,r-1} \wedge (s_{j-1,r} \wedge s_{j,r} \vee d^1_{j-1,r} \wedge d^1_{j,r} \vee d^0_{j-1,r} \wedge d^0_{j,r}) \tag{17}$$

The constraints for $gt_{j,r}$ (with $gt_{j,0} = 0$) are as follows:

$$gt_{j,r} \leftrightarrow gt_{j,r-1} \vee eq_{j,r-1} \wedge s_{j-1,r} \wedge s_{j,r} \vee eq_{j,r-1} \wedge d^1_{j-1,r} \wedge d^0_{j,r} \tag{18}$$

Finally, for the last feature, each term must be greater than the preceding one:

$$\bigwedge_{j=1}^{N} (gt_{j,K}) \tag{19}$$

4.1.4 Learning Decision Sets with SMT

After various testing with SAT Models, we decided to try a new approach. Our new model, made with SMT reasoners, had a different set of variables and constraints to work around the various distinct values in each feature. We hoped for lower constraints and size due to the innate properties of SMT but, to our surprise, our prototype didn't help at length and didn't have the best results.

The sets of variables to use for SMT Models are:

1. l_{jr}: Selects the literal on feature r for rule j, if feature r considers more than O_r different values, then $l_{jr} \in \{-O_r, ..., O_r\}$. $l_{jr} = 0$ denotes that the rule does not have a literal in feature r and $l_{jr} = -\tau$ (for $\tau \in \{1, ..., O_r\}$) denotes that rule j has a literal "feature r is not equal to τ";
2. d_{jr}^{τ}: whether rule j discriminates feature r on value τ, with $1 \leq \tau \leq O_r$ (with 0 meaning that the literal is not used);
3. cr_{jq}: whether (used) rule j covers e_q;
4. pc_j: denotes the picked class i.e the class associated with rule j, with $pc_j \in \{1, ..., L\}$.

The constraints needed in order to build the SMT Model are:

1. For each rule, at least one literal is specified meaning the sum of all literals in each rule should be greater than 0:

$$\bigvee_{r=1}^{K} l_{jr} > 0 \tag{20}$$

2. To discriminate some value τ:

$$d_{jr}^{\sigma} \leftrightarrow (l_{jr} \neq \tau \wedge l_{jr} \neq 0) \tag{21}$$

3. A rule covering some example $e_q \in \mathcal{E}$ is:

$$cr_{jq} \leftrightarrow \bigwedge_{r=1}^{K} \neg d_{j,r}^{\sigma(r,q)} \tag{22}$$

4. Every example must be covered by some rule:

$$\bigvee_{j=1}^{N} cr_{j,q}, e_q \in \mathcal{E} \tag{23}$$

5. A rule is associated with some class if and only if all examples associated with other classes are discriminated:

$$pc_j = u \leftrightarrow \bigwedge_{e_q \notin \mathcal{E}^u} \bigvee_{r=1}^{K} d_{j,r}^{\sigma(r,q)}, u \in \{1, ..., L\} \tag{24}$$

4.1.4.1 Symmetry Breaking for SMT

In order to break symmetry for our SMT model, we take the same approach as previously mentioned for SAT models. We still use the eq and gt vars and define: $eq_{j,0} = 1$ and $gt_{j,0} = 0$.

Then, with some changes in contrast to our previous Symmetry Breaking Predicates, these two next constraints that are added:

$$eq_{jr} \leftrightarrow eq_{j,r-1} \wedge (l_{j,r-1} = l_{j,r}) \tag{25}$$

$$gt_{j,r} \leftrightarrow gt_{j,r-1} \vee (l_{j,r-1} < l_{j,r}) \tag{26}$$

Finally, this constraint which is the same as (19), is also added:

$$\bigwedge_{j=1}^{N} (gt_{j,K})$$

4.2 Learning Optimal Decision Trees with SAT

The construction of an optimal decision tree is well-known to be NP-Hard [33, 34]. Practical algorithms use an heuristic approach to tackle learning decision trees. Our approach to learning a decision tree is to guess a valid binary tree topology and then verify that we can classify all positive and negative examples correctly for said topology. Therefore, the encoding consists of creating constraints that encode a valid binary tree and ensure the decision tree is accurate when classifying the examples.

4.2.1 Encoding Valid Binary Trees

In this section, the encoding of a binary tree is considered with N nodes and K binary features. Nodes are numbered with breadth-first search, from left to right. The root node is numbered 1 and the two children of a node i can be numbered from $i+1$ to $\min(2i+1, N)$. The numbers of the children are consecutive. For each node, propositional variables are added in order to encode information about the nodes. All of these are visible in 4.2. The

first four variables are used to encode a valid binary tree, while the other 5 are used to discriminate target nodes.

Variable	Description
v_i	1 iff node i is a leaf node, i=1,...,N
l_{ij}	1 iff node i has node j as the left child, with $j \in LR(i)$, where $LR(i) = even([i+1, \min(2i, N-1)]), i = 1, ..., N$
r_{ij}	1 iff node i has node j as the right child, with $j \in RR(i)$, where $RR(i) = odd([i+2, \min(2i+1, N)]), i = 1, ..., N$
p_{ji}	1 iff the parent of node j is node i, $j = 2, ..., N, i = 1, ..., N-1$
a_{rj}	1 iff feature f_r is assigned to node j, $r = 1, ..., K, j = 1, ..., N$
u_{rj}	1 iff feature f_r is discriminated against by node j, $r = 1, ..., K, j = 1, ..., N$
d^0_{rj}	1 iff feature f_r is discriminated for value 0 by node j, or by one of its ancestors, $r = 1, ..., K, j = 1, ..., N$
d^1_{rj}	1 iff feature f_r is discriminated for value 1 by node j, or by one of its ancestors, $r = 1, ..., K, j = 1, ..., N$
c_j	1 iff class of leaf node j is 1, $j = 1, ..., N$

Table 4.2: Description of propositional variables

The constraints used to encode valid binary trees are:

1. First, we assume the root node is not a leaf:

$$(\neg v_1) \tag{27}$$

2. If a node is a leaf node, then it has no children:

$$v_i \rightarrow \neg l_{ij}, j \in LR(i) \tag{28}$$

3. The left child and the right child of the ith node are numbered consecutively.

$$l_{ij} \leftrightarrow r_{ij+1}, j \in LR(i) \tag{29}$$

4. A non-leaf node must have a child.

$$\neg v_i \rightarrow \left(\sum_{j \in LR(i)} l_{ij} = 1 \right) \tag{30}$$

5. If the ith node is a parent, then it must have a child.

$$\begin{aligned} p_{ji} &\leftrightarrow l_{ij}, j \in LR(i) \\ p_{ji} &\leftrightarrow r_{ij}, j \in RR(i) \end{aligned} \tag{31}$$

6. The binary tree must be a tree which means that all nodes except the first must have a parent.

$$\left(\sum_{i=\lfloor \frac{j}{2} \rfloor}^{\min(j-1,N)} p_{ji} = 1 \right), j = 2, ..., N \tag{32}$$

Symmetry Breaking is done by ensuring the nodes in left branch will be associated with some feature being assigned value 0 and in the right branch the value 1.

4.2.2 Computing Decision Trees with SAT

Given a valid binary tree, each leaf node will be associated with the positive or the negative class. If the node's class is positive, then the path in the tree and literals associated with each branch must discriminate all the negative examples, otherwise, the classification will not be 100% accurate. If the node's class is negative, then the path in the tree and all literals must discriminate all positive examples for the same reason. This section develops constraints to achieve target discrimination and 100% accuracy while classifying all examples in $\mathcal{E}$.

To further explain discriminating targeting nodes, any example exhibiting $f_r = 0$ will be discriminated by node j or by one of its ancestors if and only if $d^0_{rj} = 1$, or $f_r = 1$ by $d^1_{rj} = 1$.

The constraints used to achieve the previous statements are:

1. To discriminate a feature for value 0 at node j, $j = 2, ..., N$:

$$d^0_{rj} \leftrightarrow \left(\bigvee_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} ((p_{ji} \wedge d^0_{ri}) \vee (a_{ri} \wedge r_{ij})) \right); d^0_{r,1} = 0 \tag{33}$$

2. To discriminate a feature for value 1 at node j, $j = 2, ..., N$:

$$d^1_{rj} \leftrightarrow \left(\bigvee_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} ((p_{ji} \wedge d^1_{ri}) \vee (a_{ri} \wedge l_{ij})) \right); d^1_{r,1} = 0 \tag{34}$$

3. Using a feature r at node j, with $r = 1, ..., K$,$j = 1, ..., N$:

$$\begin{gathered} \bigwedge_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} (u_{ri} \wedge p_{ji} \rightarrow \neg a_{rj}) \\ u_{rj} \leftrightarrow \left(a_{rj} \vee \bigvee_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} (u_{ri} \wedge p_{ji}) \right) \end{gathered} \tag{35}$$

4. For a non-leaf node j, exactly one feature is used:

$$\neg v_j \rightarrow \big(\sum_{r=1}^{K} a_{rj} = 1\big), j = 1, ..., N \tag{36}$$

5. For a leaf node j, no feature is used (only a target value $\in \{0, 1\}$):

$$v_j \rightarrow \big(\sum_{r=1}^{K} a_{rj} = 0\big), j = 1, ..., N \tag{37}$$

6. Let $e_q \in \mathcal{E}^+$, and let the sign of the literal on feature f_r for e_q be $\sigma(r, q) \in 0, 1$. For every leaf node j, $j = 1, ..., N$:

$$v_j \wedge \neg c_j \rightarrow \bigvee_{r=1}^{K} d_{r,j}^{\sigma(r,q)} \tag{38}$$

7. Let $e_q \in \mathcal{E}^-$, and let the sign of the literal on feature f_r for e_q be $\sigma(r, q)$. For every leaf node j, $j = 1, ..., N$:

$$v_j \wedge c_j \rightarrow \bigvee_{r=1}^{K} d_{r,j}^{\sigma(r,q)} \tag{39}$$

For a Decision Tree Learning problem with K binary features, $M = |\mathcal{E}|$ and a target decision tree with N nodes, the proposed encoding (on the number of literals) is in $\mathcal{O}(K \times N^2 + M \times N \times K)$. This modeling shows a far tighter encoding than the one proposed in [16], which was $\mathcal{O}(K \times N^2 \times M^2 + N \times K^2 + K \times N^3)$ and required a fixed binary tree.

Proof. (Sketch) Constraints proposed in this section have certain ranges: r goes from 1 to K, i and j go from 1 to N and the size of $\mathcal{E}$ is M. The term $M \times N$ results from Equations (38) and (39) each containing $\mathcal{O}(K)$ literals. The term $K \times N^2$ depends from the remaining constraints.

Chapter 5

Experimental Results

In this section, we intend to explain our experimental setup and compare thoroughly benchmarks on a set of 49 datasets (with the .csv format), using MinDS and other available tools.

5.1 Consistency on Datasets

It is important to note that some datasets used in our experiments were inconsistent. For example, having multiple occurrences of the same samples marked by different labels. Since our proposed models assume consistent data (in order to achieve perfect accuracy), these datasets were replaced by their largest consistent subset.

5.2 Experimental Setup

The proposed models, referenced in section 4.1.2, were implemented as a Python script, using MiniSat 2.2 SAT Solver [35] as a SAT oracle. Although these models target binary classification, most practical benchmarks nowadays require non-binary classification with features taking many distinct values. If this happens, we can use standard one-hot encoding techniques. The SMT model, referenced in section 4.1.3, was also implemented as a Python script with the PySMT library [36], using Yices2 solver [37] and Z3 solver [38] as SMT oracles.

The implementation of the two MinDS_3 models, referred in 4.1.2.1, can be generalized to MinDS_2. For name referencing, the novel encoding of MinDS_2 (alternative model) is simply named MinDS_2, while the model based on [24] will be named MP92. To test proposed symmetry breaking predicates (SBP), some models are augmented with Symmetry Breaking constraints. The models that have been augmented have the following notation: "+ SBP". As a general rule, Symmetry Breaking improves performance and reduces the

number of constraints. Finally, IDS [1] (Interpretable Decision Sets) [1], a recent approach based on Smooth Local Search [22], was also tested. IDS uses the Apriori algorithm, which identifies frequent individual items in the dataset, for the generation of candidate itemsets. The default threshold is equal to 0.2. For this setup, there was also an increase of this value to 0.5, resulting in two IDS configurations: IDS-supp0.2 and IDS-supp0.5.

These experiments were performed on a subset of the PMLB repository [2] [39]. The number of samples in the selected datasets goes from 87 to 49621 ($\approx$ 1651,1 on average), and the number of (non-binarized) features from 4 to 59 ($\approx$ 15,1 on average). Applying one-hot encoding, the number of features tends to go up, depending on the number of distinct values of each feature, from 6 to 2232 ($\approx$ 353,1 on average). The total number of datasets selected were 49. The chosen datasets can be seen in the Appendix A.

To understand how one-hot encoding works, given a non-altered dataset hayes-roth, the following features are present: Hobby, Age, Education, Marital status. After applying one-hot encoding the features are: Hobby:2.0, Hobby:1.0, Hobby:3.0, Age:2.0, Age:1.0, Age:4.0, Age:3.0, Education:2.0, Education:1.0, Education:4.0, Education:3.0, Marital status:2.0, Marital status:1.0, Marital status:4.0, Marital status:3.0.

The setup for running the MinDS models was performed in Ubuntu Linux on an Intel Xeon E5-2630 2.60GHz processor with 64GByte of memory. The time limit was set to 600 seconds and the memory limit to 10GB for each individual process to run. Another setup was run for WEKA [40] and Orange [41] on my personal computer, performed in Windows 10 on an Intel Core 5-6400 Quad-core 2.70 GHz processor with 8GByte memory.

The models used from Weka were JRIP and PRISM. These were run with the Weka package, on Java. MODLEM [42] was also considered but failed to go in accordance to our decision sets format. Even though we used binarized decision sets, MODLEM is optimized to group up certain features, reducing the amount of features. On a standard dataset, this model would work reasonably well. An example of a decision set (with 100% accuracy) for the irish.csv dataset is:

Rule 1. (Educational_level in $\{6, 5, 4, 7, 3, 9, 8, 2\}$) $\rightarrow$ (target = 0) (278/278, 100%)
Rule 2. (Educational_level in $\{10, 0, 1\}$) $\rightarrow$ (target = 1) (222/222, 100%)

Although this is an innovative way to reduce literals, we decided to stay on course and remove MODLEM from our results.

[1] `https://github.com/lvhimabindu/interpretable_decision_sets/`
[2] `https://github.com/EpistasisLab/penn-ml-benchmarks/`

CN2 Unordered Learner, which is available on the Orange library, was implemented as a Python script. Since Orange's Table, the structure necessary to prepare the data, requires alterations to the datasets, the following changes were made: on the header, each feature and target was changed to D#"feature" and cD#"target", respectively. After this, the Unordered List Learner was able to run without problems.

5.3 Scalability

The number of benchmarks solved by each model is shown in the following table (5.1):

MP92	MP92+SBP	MinDS_2	MinDS_2+SBP	MinDS_1	MinDS_1 + SBP	IDS-Supp 0.2	IDS-Supp 0.5
42	45	42	45	6	6	0	2

Table 5.1: Number of solved instances per model (out of 49)

As said previously, the binarization of features means a larger number of features (K). Despite that, MinDS_2 and MP92 still solved a considerable number of datasets, 42 out of 49. Symmetry Breaking also brings significant improvements, solving three more instances than their non-SBP counterparts. MinDS_1 and MinDS_1+SBP don't perform as well as previous models, probably due to the fact that they target decision sets without $\mathcal{U}^{\ominus}$-overlap, which are significantly harder to solve.

To our surprise, the performance of IDS was poor in practice, being unable to solve any considerable instances within the 600 seconds threshold. Although IDS aims to maximize the number of covered training samples and to minimize rule overlap, the rules produced usually exhibit significant overlap. Seeing as IDS performs poorly, most attention will be paid to MinDS, Weka and Orange models.

In my alternate setup, the 3 models were able to produce all the decision sets in respect to each dataset. Following the previous setup of 600 seconds, the Unordered List Learner from the Orange library only failed 1 dataset (this one being the diabetes dataset) while the other models were able to build most datasets in under 100 seconds.

Weka - JRIP	Weka - Prism	Orange - CN2 Unordered List Learner
49	49	48

Table 5.2: Number of solved instances per model (out of 49) - Alternate setup

5.4 Assessing Quality

The proposed SAT models target minimizing the number of rules, not the number of literals, in their decision sets. Quality of decision sets is subjective, some people might prefer fewer rules with more literals while others might prefer a larger number of rules with few literals as possible. As soon as the number of rules in the decision set is minimized, we can try to minimize the number of literals in the resulting decision sets, by applying applying a Boolean Lexicographic Optimization [43]. A simple MaxSAT problem can be devised by augmenting the formula with the unit soft clauses, which force all literals of the decision set to be unused. Afterwards, the minimum number of literals can be computed by a standalone MaxSAT solver or approximated with the use of an MCS enumerator [20]. While the former approach is exact, it is often outperformed by the latter. For more information on MCSes, it is advised to read 2.4 thoroughly. With that being said, the configuration marked by A10 in the Appendix A and in the next section indicates that 10 MCSes were computed to approximate solutions for the literal minimization problem devised.

5.5 Benchmark comparison

In order to assess performance and quality, we're going to compare the benchmarks of various models.

In order to test accuracy, an approach was taken to test on the same training dataset. Even though this is usually not recommended, we did the same for Weka/Orange in order to maintain the integrity of the setup. Some k-cross-validation tests (which are usually the preferred method in Machine Learning) were also run on Weka and Orange, for the sake of checking. All SAT and SMT models use Symmetry Breaking Predicates due to how effective they proved to be in Table 5.1.

Assessing JRIP This model was run with default parameters with the exception of pruning, which was removed in order to improve accuracy. With pruning, we would get worse results in terms of accuracy and sometimes rules similar to: "If true then class 1". The interpretability in rules such as these is non-existent. The average accuracy on all datasets reaches around 88,43%, which is acceptable but not as good as other models. Some datasets can only reach 42,6%, which is worse than a coin flip. Between all datasets, the build time averages around 0.52 seconds, 16 rules and finally, 65 literals. This model also solves all 49 datasets. Performing a 10 cross-validation test proves to be within around the same accuracy of the test on the same training dataset, proving that this model is reliable, non-dependant on the training dataset to build models and finally, doesn't overfit. Even though accuracy isn't the best, JRIP compensates by having a small number of rules,

literals and a reduced model building time, so we can expect small decision sets that are interpretable to a non-expert human, done in an efficient time lapse.
The full benchmarks can be found on the Appendix, in A.1.

Assessing PRISM This model's accuracy is definitely better than JRIP, with only 2 datasets not reaching 100% accuracy. PRISM averages 99,72% accuracy with the build time averaging around 2.85 seconds (with only 5 datasets taking longer than 1 second). The problem with this model revolves around the huge number of rules and literals, averaging 147 and 532, respectively. This is nowhere near acceptable on an interpretability standpoint. Even though the accuracy, on the same training dataset, and time is surprising, it falls short of desirable on the rules and literals. Another con is that if we perform a standard 10-cross-validation, the accuracy suddenly drops and has too many unclassified instances, proving that this model is nowhere near as good performance-wise on a k-1 fold partition of the same dataset as it is on the same training dataset. This model might overfit, relying too much on the training data and not working on new unknown data. The model solves all 49 datasets.
The full benchmarks can be found on the Appendix, in A.2.

Assessing CN2 This model, CN2 Unordered List Learner from Orange, was run with default parameters. This model does not try to minimize rule overlap as our models do. This model has solid accuracy, with 7 datasets not reaching 100%, but always near it. CN2 averages an accuracy of 99,79% and build time of 99.1 seconds. In comparison with PRISM, CN2 steps up with an average of 109 rules and 224 literals. Even though it takes more time, it might be preferable to run a model that has a lower quantity of rules and literals, in order to provide a smaller, more interpretable decision set. This model solves 49 datasets or, if taken into account the 600 seconds time limit, 48.
The full benchmarks can be found on the Appendix, in A.3.

The next 4 models (MP92, MP92 + A10, MinDS_2, MinDS_2 + A10) solve 45 out of 49 models. In order to compare benchmarks, the average of the parameters was done with the same 45 datasets solved for JRIP, PRISM and CN2. These 4 unsolved models (balance-un, breast-cancer-un, contraceptiveM-un and spect-un) were the most intensive, whether time or rule/literal wise. The average for each model, in respect to rules, literals, accuracy, and time are:

- JRIP - 16, 65, 89,24%, 0.56
- PRISM - 126, 287, 99,81%, 0.65
- CN2 - 93, 159, 99,87%, 102.13

Assessing MP92 Model This model, based on [24], solves 45 out of 49 models. Since our models focus on lowering the number of rules, the average number of rules is $\approx$ 12. The problem is, without the additional computing of 10 MCSes, the number of literals is too high, with an average of 886 literals. Despite the large models (literal wise), there is still an acceptable average time on model build time, taking $\approx$ 16.7 seconds. As previously mentioned, an average of 886 literals is too high since there can be no interpretability if we lose ourselves reading a rule that has too many literals.
The full benchmarks can be found on the Appendix, in A.4.

Assessing MP92 with A10 This model, based on [24], with the computation of 10 MCSes, solves 45 models out of 49 models. Since the computing of Minimal Correction Subsets only helps after the rules have been minimized, the average number of rules stay the same as before, with the average of 12. The number of literals decreased noticeably, reducing in size by $\approx$ 72%, to 252 literals. The computation of 10 MCSes also adds extra build time, increasing the average time to 20.6 seconds. In comparison with PRISM and CN2, MP92+A10 fairs pretty well, beating accuracy, rules, and literals. It also beats CN2 on time.
The full benchmarks can be found on the Appendix, in table A.5.

Assessing MinDS$_2$ This model, proposed with different variables and additional clauses, solves 45 out of 49 models. The average number of rules is approximately 12 and in comparison to the previous MP92 model without MCSes it has $\approx$ 47% fewer literals, averaging 470 literals. In contrast to the MP92's average time of 16.7 seconds, this model takes on average 22.8 seconds to build a model.
The full benchmarks can be found on the Appendix, in A.6.

Assessing MinDS$_2$ with A10 This model, with the computation of 10 MCSes, solves 45 out of 49 models. The number of rules stays the same, with the average of 12, meanwhile the number of literals drop to 250, which is a 47% decrease in comparison to the benchmarks without MCSes. The build time adds some seconds, averaging 29 seconds with the computation of 10 MCSes. Same as MP92 with A10, it outperforms PRISM and CN2 on rules, literals, and accuracy. Also beats CN2 on time.
The full benchmarks can be found on the Appendix, in A.7.

Assessing MinDS$_1$ MinDS$_1$ didn't perform as well as other models due to the fact that it targets decision sets without $\mathcal{U}^{\ominus}$-overlap, which makes them harder to solve. Meanwhile, previous models had an easier time solving $\mathcal{E}$-overlap. MINDS$_1$ only solved 6 datasets and also had a slight increase in the number of rules in comparison to previous models such as MINDS$_2$ and MP92. There is also a large number of literals since it lacks the

computation of 10 MCSes.
The full benchmarks can be found on the Appendix, in A.8.

Assessing MinDS$_1$ with A10 The benchmarks between MinDS$_1$ and MinDS$_1$ with A10 are similar. Only 6 datasets were solved but computing 10 MCSes helps reducing the number of literals. There is still an increase in the number of rules and literals in comparison to models such as MinDS$_2$ and MP92. It is to be expected since this model targets $\mathcal{U}^{\ominus}$-overlap.
The full benchmarks can be found on the Appendix, in A.9.

Assessing SMT This SMT model, which was trialed and prototyped later, had an objective to work around the distinct values of each features. With either the Yices2 [37] or Z3 [38] solver, the SMT approach only managed to solve 9 datasets. There is also a slight increase in the number of rules and literals in comparison to previous models.
The full benchmarks can be found on the Appendix, for Z3 and Yices2, respectively, in A.10 and in A.11

In summary, our models had interesting results which bring a positive overview of this logic approach to machine learning. Since only 4 datasets could not be solved by our MP92/MinDS$_2$ variants, I decided to take the benchmarks from JRIP, PRISM and CN2 in order to compare benchmarks between models more efficiently. As previously mentioned our models target term (rule) minimization, so we can always expect a lower amount of rules.

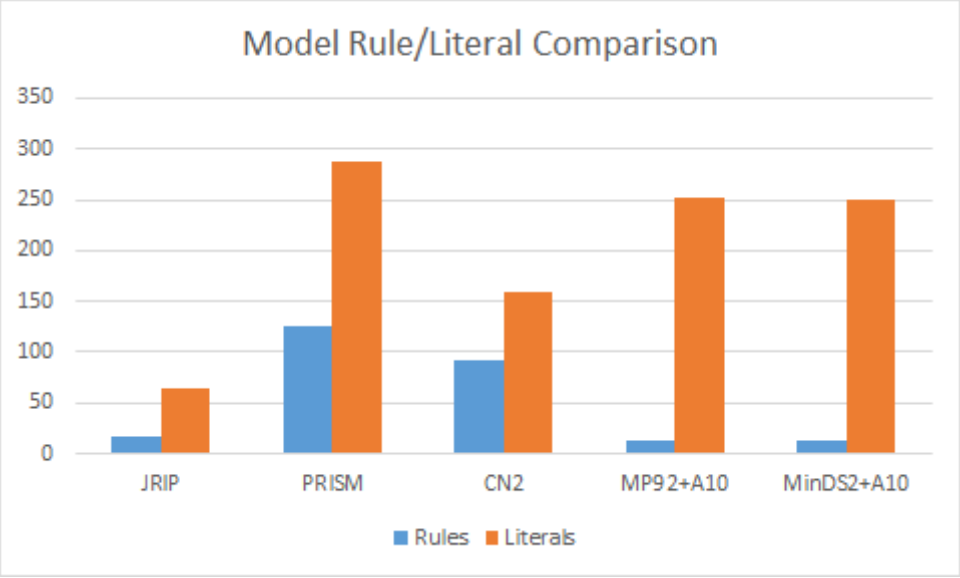

Figure 5.1: Model Rule and Literals Comparison

Graph 5.1 shows the comparison between rules and literals of each model. The av-

erage amount of rules and literals is 52 and 203, respectively. JRIP has a low amount of rules and literals, which could make this model the most interpretable one but since it does not have decent accuracy, it would be advised to steer clear from it if we value precision. If our purpose was to bring a quick and easy decision set to the mix, this could be the model to pick. On the other hand, PRISM has a huge amount of rules and literals with a bigger accuracy, with the same time taken to build the model. CN2 has a lower amount of rules and literals than PRISM, but takes more time in order to reach almost perfect accuracy. PRISM would also fail if not tested on the same training dataset, whereas CN2 and JRIP would not. Finally, MP92+A10 and MinDS_2+A10 show the least amount of rules with a slightly higher amount of literals than the average (250/252, respectively, versus $\approx$ 203 literals). As discussed in the following graph, the time it took to build these models is completely acceptable for what our models bring to the table.

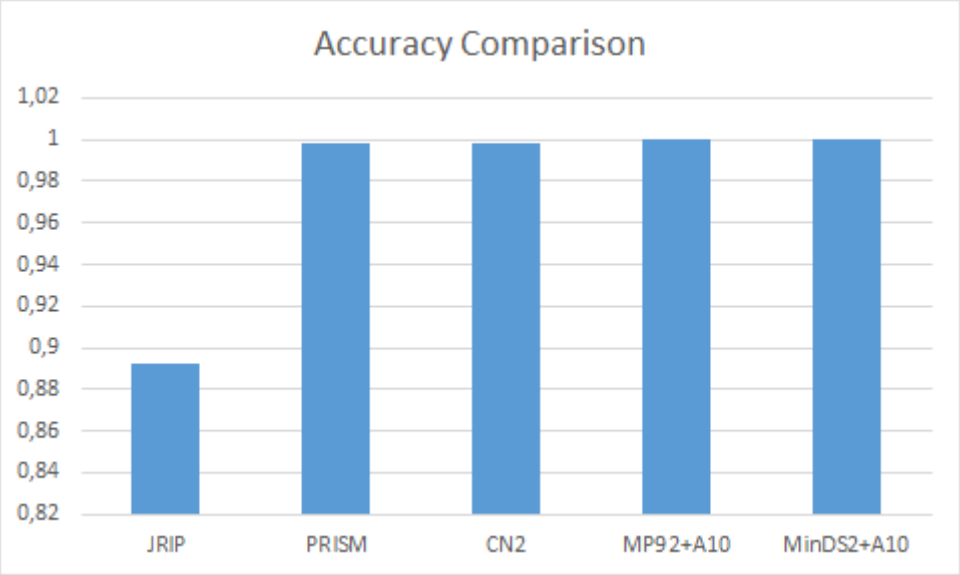

Figure 5.2: Accuracy comparison

Graph 5.2 shows the comparison of accuracy between models. JRIP fairs well in performance but not in accuracy, which would be its biggest drawback. While JRIP achieves 89%, all the other models reach 99,5% accuracy or higher. Meanwhile our MinDS models reach 100%. At what point do we trade smaller, more interpretable decision sets in order to attain better accuracy. At what point is performance favored over accuracy?

Graph 5.3, shows the comparison of time between all models which averages $\approx$ 30.6 seconds. JRIP and PRISM average less than a second to build their model. On the other hand, CN2, MP92 and MinDS_2 take longer than their counterparts. CN2, with the average of 102 seconds, doesn't compare to the other models. This is a huge amount of time in comparison to MP92 and MinDS_2 which average 20.6 and 29 seconds, respectively. If time matters CN2, even though almost 100% accurate, is not the ideal model. Even

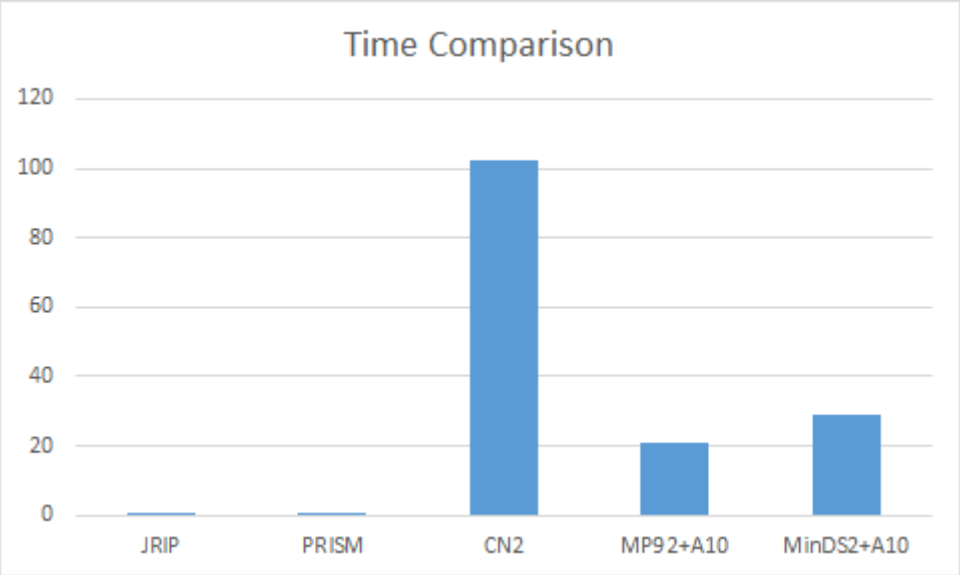

Figure 5.3: Time comparison (in seconds)

though it can be run on almost any desktop or laptop with a decent CPU and RAM, its large number of rules and literals is also a drawback.

5.6 Additional Testing

For early testing of MinDS models, a script was created that read a DIMACS DNF format file and generated datasets based on that DNF expression, adding dummy variables (total number of variables could amount up to 8, 16 and 32 in total) in order to find out if our models could find the correct variables to use. This small exercise was based on earlier work [24] and the models usually took the correct approach, identifying the same disjunctive normal form present in the original file. Some problems could be found on datasets with 250 samples and 32 variables but could easly solve any other dataset at ease.

Chapter 6

Discussion

This chapter provides a summary of the thesis, some final conclusions, the new European Union Regulations for data collection [7] and a brief overview of future work.

6.1 Summary of Thesis

In Chapter 3 we brought a quick overview of what is available in XAI and its two main areas were categorized, these being Interpretable Models and Prediction Interpretation and Justification. The first area is of utmost importance due to the connections made to our models. The second area is also crucial due to the frameworks presented, bringing interpretability to Neural Networks, the most accurate and efficient of black box models. The Smallest Decision Tree Problem is also studied, in which we explore the SAT-Based encoding to minimize tree nodes and study the Space Complexity to further justify our approach. In Chapter 4 there is a deep analysis of the work done throughout the thesis. Earlier work paved the road to Decision Sets, introducing definitions for itemsets, rules, and overlap. MINDS optimization problems are presented and proven to be NP-hard. For each problem, this thesis proposes a SAT model that enables finding optimum solutions in the number of terms, with its respective variables and encodings. Symmetry Breaking Predicates are also studied due to them helping reduce the complexity of our problem. A SMT model and its respective symmetry breaking predicates are also prototyped but failed to bring decent results. Learning Optimal Decision Trees with SAT is also mentioned, in which the main objective is to learn a decision tree to guess a valid binary tree topology and then, verifying whether or not it can classify all positive and negative examples correctly, ensuring the decision tree is accurate. In Chapter 5 our setups are explained and benchmarks are compared with other available tools. Our 49 datasets are binarized with standard one-hot encoding techniques and made consistent. We prove that JRIP is a fast tool with low number of rules and literals but not accurate while PRISM and CN2 have a large number of rules and literals, but are more precise. Our MP92 and MINDS$_2$ approaches manage to ensure the lower amount of rules possible and through the com-

putation of 10 Minimal Correction Subsets, there is a large decrease in the number of literals.

6.2 Final Conclusions

As proved in sections 4 and 5, decision set learning through SAT proves to be a viable option since these decision sets represent a promising and interpretable approach to provide explanations in different Machine Learning setting. The problem lies in learning optimal decision sets (which is proven to be NP-Hard) and their difficulties regarding overlap. MINDS formulation optimization mentioned in this paper is mostly hard for NP but ensuring decision sets with no feature space overlap is in Σ_2^p.

6.3 European Union Regulations

In April 2016, for the first time in over two decades, the European Parliament adopted a set of comprehensive regulations for the collection, storage and use of personal information, the General Data Protection Regulation (GDPR), in order to give control back to citizens and residents regarding how their personal data is acquired, stored, secured and processed. This new regulation, which is already taking effect as a law across Europe in 2018, will restrict automated individual decision-making which affects users. This new law will also create a ”right to explanation” where a user can ask for an explanation of an algorithmic decision that was made about them, thus, the importance of Explainable Artificial Intelligence.

This regulation contains *Article 22: Automated Individual Decision-Making, including profiling* potentially prohibiting a wide variety of algorithms currently in use (from recommendation systems, credit and insurance risk assessments, computational advertising to social networks) so in a way, it will bring a complete overhaul of standard and widely used algorithmic techniques.

Taken from [7], the following quote ”Big Data claims to be neutral. It isn’t.” has a large resonance. If we have large datasets where a group is underrepresented there is more uncertainty associated with predictions in relation to that group, which means data is inherently discriminatory.

The right to an explanation and its provisions are outlined in *Articles 13-15*, which specify that data subjects have the right to access information collected about them and also requires data processors to ensure data subjects are notified about the data collected. *Articles 13-14* also state that when profiling takes place a data subject has the right to “meaningful information about the logic involved”. So the right question is, what is required to explain an algorithm’s decision?

Due the time this Thesis is submitted, the legislation and regulations will already be in effect (since May 25th) with major companies like Facebook, Google, Twitter and others providing new Terms of Service.

6.4 Future Work

The experimental results show relevance of the SAT-based learning of optimal decision sets, so one of the goals is to provide and devise new Machine Learning Models based on logic reasoners, whose objective is to associate explanations with predictions while producing viable models with good accuracy and performance.

These results also motivate the possibility of developing more efficient propositional encodings and to consider new approaches to learning optimal decision sets. Further study should also be done to find a viable way to perceive interpretability within the total size of the decision set, either by minimizing the total number of literals instead of rules or multi-objective optimization. Since this approach can result in smaller and better interpretable solutions, further comparisons can be made to heuristic rule-based classifiers.

Bibliography

[1] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1675–1684, ACM, 2016.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice Hall Press, 3rd ed., 2009.

[3] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[4] B. Letham, C. Rudin, T. H. McCormick, D. Madigan, *et al.*, "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model," *The Annals of Applied Statistics*, vol. 9, no. 3, pp. 1350–1371, 2015.

[5] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin, "Learning Certifiably Optimal Rule Lists," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 35–44, ACM, 2017.

[6] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep Learning for Case-based Reasoning through Prototypes: A Neural Network that Explains its Predictions," *arXiv preprint arXiv:1710.04806*, 2017.

[7] B. Goodman and S. Flaxman, "European Union regulations on algorithmic decision-making and a" right to explanation"," *arXiv preprint arXiv:1606.08813*, 2016.

[8] NIPS IML Symposium, "NIPS interpretable ML symposium," 2017.

[9] ICML WHI Workshop, "ICML workshop on human interpretability in ML," 2017.

[10] IJCAI XAI Workshop, "IJCAI workshop on explainable artificial intelligence," 2017.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[12] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[13] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," *arXiv preprint arXiv:1708.08296*, 2017.

[14] A. Ignatiev, F. Pereira, N. Narodytska, and J. Marques-Silva, "A SAT-Based Approach to Learn Explainable Decision Sets." in IJCAR, preprint available in: `https://reason.di.fc.ul.pt/~aign/publ/ipnms-ijcar18-preprint.pdf`, 2018.

[15] N. Narodytska, A. Ignatiev, F. Pereira, and J. Marques-Silva, "Learning Optimal Decision Trees with SAT." in IJCAI, preprint available in: `https://reason.di.fc.ul.pt/~aign/publ/nipms-ijcai18-preprint.pdf`, 2018.

[16] C. Bessiere, E. Hebrard, and B. O'Sullivan, "Minimising decision tree size as combinatorial optimisation," in *International Conference on Principles and Practice of Constraint Programming*, pp. 173–187, Springer, 2009.

[17] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh, *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2009.

[18] M. Sipser, *Introduction to the Theory of Computation*. Course Technology, second ed., 2006.

[19] A. Morgado, F. Heras, M. Liffiton, J. Planes, and J. Marques-Silva, "Iterative and core-guided MaxSAT solving: A survey and assessment," *Constraints*, vol. 18, no. 4, pp. 478–534, 2013.

[20] J. Marques-Silva, F. Heras, M. Janota, A. Previti, and A. Belov, "On Computing Minimal Correction Subsets.," in *IJCAI*, pp. 615–622, 2013.

[21] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI-17 Workshop on Explainable AI (XAI)*, p. 8, 2017.

[22] U. Feige, V. S. Mirrokni, and J. Vondrak, "Maximizing non-monotone submodular functions," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 1133–1153, 2011.

[23] F. Wang and C. Rudin, "Falling rule lists," in *Artificial Intelligence and Statistics*, pp. 1013–1022, 2015.

[24] A. P. Kamath, N. K. Karmarkar, K. Ramakrishnan, and M. G. Resende, "A continuous approach to inductive inference," *Mathematical programming*, vol. 57, no. 1-3, pp. 215–238, 1992.

[25] Y. Lou, R. Caruana, and J. Gehrke, "Intelligible models for classification and regression," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 150–158, ACM, 2012.

[26] J. Clos, N. Wiratunga, and S. Massie, "Towards Explainable Text Classification by Jointly Learning Lexicon and Modifier terms," in *IJCAI-17 Workshop on Explainable AI (XAI)*, p. 19.

[27] W. W. Cohen, "Fast Effective Rule Induction," in *Twelfth International Conference on Machine Learning*, pp. 115–123, Morgan Kaufmann, 1995.

[28] J. Cendrowska, "PRISM: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, no. 4, pp. 349–370, 1987.

[29] P. Clark and R. Boswell, "Rule induction with CN2: Some recent improvements," in *European Working Session on Learning*, pp. 151–163, Springer, 1991.

[30] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, ACM, 2016.

[31] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & Explorable Approximations of Black Box Models," *arXiv preprint arXiv:1707.01154*, 2017.

[32] R. L. Rivest, "Learning decision lists," *Machine learning*, vol. 2, no. 3, pp. 229–246, 1987.

[33] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is NP-complete," *Information processing letters*, vol. 5, no. 1, pp. 15–17, 1976.

[34] T. Hancock, T. Jiang, M. Li, and J. Tromp, "Lower bounds on learning decision lists and trees," *Information and Computation*, vol. 126, no. 2, pp. 114–122, 1996.

[35] N. Eén and N. Sörensson, "An extensible SAT-solver," in *International conference on theory and applications of satisfiability testing*, pp. 502–518, Springer, 2003.

[36] M. Gario and A. Micheli, "PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms," in *Proceedings of the 13th International Workshop on Satisfiability Modulo Theories (SMT)*, pp. 373–384, 2015.

[37] B. Dutertre, "Yices 2.2," in *Computer Aided Verification* (A. Biere and R. Bloem, eds.), (Cham), pp. 737–744, Springer International Publishing, 2014.

[38] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, (Berlin, Heidelberg), pp. 337–340, Springer-Verlag, 2008.

[39] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "PMLB: a large benchmark suite for machine learning evaluation and comparison," *BioData Mining*, vol. 10, p. 36, Dec 2017.

[40] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[41] J. Demšar, T. Curk, A. Erjavec, Črt Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, and B. Zupan, "Orange: Data Mining Toolbox in Python," *Journal of Machine Learning Research*, vol. 14, pp. 2349–2353, 2013.

[42] J. Stefanowski, "On combined classifiers, rule induction and rough sets," in *Transactions on rough sets VI*, pp. 329–350, Springer, 2007.

[43] J. Marques-Silva, J. Argelich, A. Graça, and I. Lynce, "Boolean lexicographic optimization: algorithms & applications," *Annals of Mathematics and Artificial Intelligence*, vol. 62, no. 3-4, pp. 317–343, 2011.

Appendix A

Benchmarks

A.1 Weka Model - JRIP Algorithm

File	Rules	Literals	Accuracy	Time (seconds)
appendicitis-un.csv	4	3	87.74%	0.221
australian-un.csv	11	56	88.41%	1.081
auto-un.csv	25	53	95.5%	0.369
backache-un.csv	8	21	98.33%	0.062
balance-un.csv	28	125	86.72%	0.02
biomed-un.csv	8	25	83.25%	0.113
breast-cancer-un.csv	10	39	88.21%	0.015
bupa-un.csv	7	27	69.57%	0.098
cars-un.csv	16	18	99.23%	0.262
cleveland-nominal-un.csv	13	69	84.62%	0.006
cleveland-un.csv	26	87	81.52%	0.16
cleve-un.csv	12	44	91.75%	0.103
cloud-un.csv	8	8	42.59%	0.05
colic-un.csv	13	48	94.57%	0.121
contraceptiveM-un.csv	12	68	46.98%	0.156
corral-un.csv	4	7	100.0%	0.003
dermatology-un.csv	12	31	98.91%	0.048
diabetes-un.csv	4	17	67.45%	0.813
ecoli-un.csv	33	66	77.37%	0.174
features_10_train.csv	226	1628	98.84%	4.537
flags-un.csv	1	1	98.88%	0.01
glass2-un.csv	19	26	87.12%	0.1
glass-un.csv	16	38	71.22%	0.193
haberman-un.csv	8	35	85.0%	0.026
hayes-roth-un.csv	11	21	100.0%	0.005
heart-c-un.csv	12	44	91.75%	0.1
heart-h-un.csv	8	35	89.12%	0.072
heart-statlog-un.csv	13	44	96.3%	0.076
hepatitis-un.csv	7	21	96.77%	0.039
house-votes-84-un.csv	10	35	99.31%	0.146
hungarian-un.csv	8	35	89.12%	0.074
irish-un.csv	4	3	100%	0.027
iris-un.csv	10	21	97.33%	0.049
liver-disorder-un.csv	7	27	69.57%	0.112
lupus-un.csv	5	6	68.6%	0.01
lymphography-un.csv	9	21	98.65%	0.008
molecular-biology_promoters-un.csv	5	9	98.11%	0.036
mux6-un.csv	5	12	100.0%	0.003
new-thyroid-un.csv	14	23	89.3%	0.053
postoperative-patient-data-un.csv	6	16	89.2%	0.003
promoters-un.csv	5	9	98.11%	0.023
schizo-un.csv	5	19	60.88%	0.02
shuttleM-un.csv	11	26	99.92%	15.599
soybean-un.csv	39	98	98.22%	0.107
spect-un.csv	6	31	95.22%	0.006
tae-un.csv	25	59	85.62%	0.018
titanic-un.csv	3	4	100.0%	0.026
uci_mammo_data-un.csv	8	26	99.11%	0.034
zoo-un.csv	8	15	99.1%	0.009

Table A.1: Weka - JRIP Benchmarks

A.2 Weka Model - PRISM Algorithm

File	Rules	Literals	Accuracy	Time (seconds)
appendicitis-un.csv	56	56	100%	0.27
australian-un.csv	456	530	100%	4.552
auto-un.csv	79	80	100%	0.294
backache-un.csv	87	87	100%	0.092
balance-un.csv	213	1066	100%	0.031
biomed-un.csv	117	118	100%	0.142
breast-cancer-un.csv	118	352	100%	0.014
bupa-un.csv	195	300	100%	0.141
cars-un.csv	30	30	100%	0.153
cleveland-nominal-un.csv	71	289	100%	0.006
cleveland-un.csv	214	305	100%	0.167
cleve-un.csv	181	235	100%	0.135
cloud-un.csv	98	98	100%	0.058
colic-un.csv	197	289	100%	0.19
contraceptiveM-un.csv	911	11345	95.11%	0.929
corral-un.csv	8	18	100%	0.004
dermatology-un.csv	74	148	100%	0.059
diabetes-un.csv	481	559	100%	8.017
ecoli-un.csv	184	275	100%	0.166
features_10_train.csv	705	5811	91.304%	12.167
flags-un.csv	7	9	100%	0.01
glass2-un.csv	95	95	100%	0.091
glass-un.csv	148	148	100%	0.201
haberman-un.csv	171	319	100%	0.031
hayes-roth-un.csv	23	65	100%	0.003
heart-c-un.csv	181	235	100%	0.134
heart-h-un.csv	189	242	100%	0.111
heart-statlog-un.csv	168	211	100%	0.109
hepatitis-un.csv	73	79	100%	0.037
house-votes-84-un.csv	34	117	100%	0.148
hungarian-un.csv	189	242	100%	0.111
irish-un.csv	11	11	100%	0.027
iris-un.csv	37	47	100%	0.036
liver-disorder-un.csv	195	300	100%	0.154
lupus-un.csv	76	83	100%	0.013
lymphography-un.csv	39	90	100%	0.007
molecular-biology_promoters-un.csv	92	92	100%	0.032
mux6-un.csv	8	24	100%	0.003
new-thyroid-un.csv	92	94	100%	0.058
postoperative-patient-data-un.csv	32	97	100%	0.005
promoters-un.csv	92	92	100%	0.033
schizo-un.csv	223	241	100%	1.3
shuttleM-un.csv	261	375	100%	109.247
soybean-un.csv	99	319	100%	0.11
spect-un.csv	51	194	100%	0.008
tae-un.csv	85	132	100%	0.014
titanic-un.csv	4	7	100%	0.019
uci_mammo_data-un.csv	30	79	100%	0.022
zoo-un.csv	15	33	100%	0.008

Table A.2: Weka - PRISM Benchmarks

A.3 CN2 - Unordered List Learner

File	Rules	Literals	Accuracy	Time(seconds)
appendicitis-un	55	55	100%	42.66
australian-un	268	345	100%	482.65
auto-un	75	78	100%	136.52
backache-un	62	67	100%	46.97
balance-un	196	704	100%	20.15
biomed-un	96	103	100%	107.74
breast-cancer-un	92	279	100%	15.32
bupa-un	155	260	100%	108.96
cars-un	31	31	100%	60.97
cleve-un	161	215	100%	109.46
cleveland-nominal-un	68	260	97.44%	5.32
cleveland-un	209	303	100%	136.97
cloud-un	99	98	100%	58.45
colic-un	142	228	100%	122.49
contraceptiveM-un	811	2724	94.98%	219.18
corral-un	7	12	100%	0.24
dermatology-un	57	124	100%	31.18
diabetes-un	373	474	100%	787.14
ecoli-un	165	251	100%	122.5
features_10_train	256	1275	98.35%	165.88
flags-un	6	8	100%	2.25
glass-un	140	140	100%	162.37
glass2-un	96	95	100%	96.47
haberman-un	125	251	100%	26.9
hayes-roth-un	22	59	100%	1.6
heart-c-un	161	215	100%	108.12
heart-h-un	116	159	100%	61.97
heart-statlog-un	140	185	100%	90.77
hepatitis-un	51	57	100%	32.45
house-votes-84-un	29	98	99.77%	2.51
hungarian-un	116	159	100%	65.15
iris-un	29	39	100%	11.12
irish-un	12	11	100%	6.72
liver-disorder-un	155	260	100%	101.41
lupus-un	77	83	100%	11.67
lymphography-un	33	74	100%	7.01
molecular-biology_promoters-un	31	37	100%	23.47
mux6-un	9	24	100%	0.34
new-thyroid-un	78	83	100%	51.13
postoperative-patient-data-un	28	83	98.78%	2.62
promoters-un	31	37	100%	21.46
schizo-un	87	86	100%	585.16
shuttleM-un	146	232	100%	526.24
soybean-un	84	273	99.7%	53.21
spect-un	39	130	97.61%	5.01
tae-un	80	125	100%	14.84
titanic-un	5	7	100%	0.16
uci_mammo_data-un	27	71	100%	1.59
zoo-un	11	22	100%	0.97

Table A.3: Orange - Unordered List Learner Benchmarks

A.4 MinDS - MP92

File	Rules	Literals	Time (seconds)
appendicitis-un	2	497	0.396
australian-un	4	2389	13.668
auto-un	5	3398	1.868
backache-un	2	360	0.564
balance-un			
biomed-un	3	1037	1.98
breast-cancer-un			
bupa-un	6	757	3.784
cars-un	3	1247	2.064
cleveland-nominal-un	52	415	122.7
cleveland-un	9	1966	2.864
cleve-un	4	669	1.94
cloud-un	4	1727	0.572
colic-un	7	906	6.908
contraceptiveM-un			
corral-un	6	14	0.072
dermatology-un	10	892	1.688
diabetes-un	4	2391	16.608
ecoli-un	9	1802	2.704
features_10_train	167	1200	153.4
flags-un	4	120	0.436
glass2-un	2	574	0.764
glass-un	5	3334	1.744
haberman-un	13	701	12.768
hayes-roth-un	17	115	0.42
heart-c-un	4	678	1.96
heart-h-un	6	813	3.356
heart-statlog-un	4	660	1.684
hepatitis-un	4	620	1.008
house-votes-84-un	20	91	34.7
hungarian-un	6	870	3.276
irish-un	2	38	0.36
iris-un	5	332	0.336
liver-disorder-un	6	744	3.712
lupus-un	4	276	0.248
lymphography-un	12	210	2.228
molecular-biology_promoters-un	2	107	0.256
mux6-un	8	24	0.084
new-thyroid-un	4	796	0.98
postoperative-patient-data-un	20	164	61.368
promoters-un	2	107	0.26
schizo-un	2	1888	5.8
shuttleM-un	5	1223	257.936
soybean-un	38	3010	15.048
spect-un			
tae-un	10	473	0.628
titanic-un	4	11	0.056
uci_mammo_data-un	18	114	2.72
zoo-un	9	110	0.08

Table A.4: MinDS – MP92 Benchmarks

A.5 MinDS - MP92 with A10

File	Rules	Literals	Time(seconds)
appendicitis-un	2	73	0.492
australian-un	4	569	31.352
auto-un	5	174	2.356
backache-un	2	150	0.792
balance-un			
biomed-un	3	220	3.356
breast-cancer-un			
bupa-un	6	366	6.308
cars-un	3	138	2.424
cleveland-nominal-un	52	317	126.04
cleveland-un	9	674	5.616
cleve-un	4	299	3.716
cloud-un	4	319	1.088
colic-un	7	236	9.648
contraceptiveM-un			
corral-un	6	12	0.08
dermatology-un	10	134	2.22
diabetes-un	4	863	52.82
ecoli-un	9	681	5.396
features_10_train	167	1148	179.368
flags-un	4	11	0.488
glass2-un	2	138	1.02
glass-un	5	686	3.612
haberman-un	13	423	16.708
hayes-roth-un	17	83	0.508
heart-c-un	4	297	3.612
heart-h-un	6	370	5.784
heart-statlog-un	4	256	2.952
hepatitis-un	4	95	1.492
house-votes-84-un	20	89	36.26
hungarian-un	6	312	5.336
irish-un	2	11	0.396
iris-un	5	123	0.48
liver-disorder-un	6	346	6.352
lupus-un	4	114	0.388
lymphography-un	12	83	2.712
molecular-biology_promoters-un	2	92	0.292
mux6-un	8	24	0.096
new-thyroid-un	4	206	1.4
postoperative-patient-data-un	20	100	59.5
promoters-un	2	92	0.3
schizo-un	2	86	6.704
shuttleM-un	5	250	316.648
soybean-un	38	301	17.708
spect-un			
tae-un	10	271	1.14
titanic-un	4	7	0.06
uci_mammo_data-un	18	82	2.88
zoo-un	9	23	0.092

Table A.5: MinDS – MP92 with A10 Benchmarks

A.6 MinDS - MinDS$_2$

File	Rules	Literals	Time (seconds)
appendicitis-un	2	73	0.364
australian-un	4	1587	12.988
auto-un	5	183	1.848
backache-un	2	180	0.528
balance-un			
biomed-un	3	531	1.872
breast-cancer-un			
bupa-un	6	727	3.496
cars-un	3	188	1.876
cleveland-nominal-un	52	401	77.444
cleveland-un	9	1329	2.832
cleve-un	4	500	1.86
cloud-un	4	324	0.556
colic-un	7	935	6.44
contraceptiveM-un			
corral-un	6	15	0.072
dermatology-un	10	407	1.628
diabetes-un	4	1612	16.132
ecoli-un	9	1116	2.636
features_10_train	167	1200	468.268
flags-un	4	42	0.408
glass2-un	2	138	0.712
glass-un	5	692	1.732
haberman-un	13	707	7.876
hayes-roth-un	17	113	0.432
heart-c-un	4	512	1.856
heart-h-un	6	771	2.992
heart-statlog-un	4	506	1.612
hepatitis-un	4	251	0.968
house-votes-84-un	20	93	51.32
hungarian-un	6	769	3.028
irish-un	2	11	0.336
iris-un	5	242	0.352
liver-disorder-un	6	703	3.536
lupus-un	4	220	0.24
lymphography-un	12	147	2.564
molecular-biology_promoters-un	2	106	0.244
mux6-un	8	24	0.084
new-thyroid-un	4	280	0.92
postoperative-patient-data-un	20	158	84.688
promoters-un	2	106	0.244
schizo-un	2	86	5.372
shuttleM-un	5	1055	233.336
soybean-un	38	1413	15.704
spect-un			
tae-un	10	540	0.624
titanic-un	4	9	0.056
uci_mammo_data-un	18	110	4.012
zoo-un	9	43	0.084

Table A.6: MINDS - SAT Benchmarks

A.7 Minds - SAT with A10

File	Rules	Literals	Time(seconds)
appendicitis-un	2	71	0.46
australian-un	4	686	46.284
auto-un	5	175	2.304
backache-un	2	137	0.76
balance-un			
biomed-un	3	218	4.24
breast-cancer-un			
bupa-un	6	366	6.816
cars-un	3	147	2.388
cleveland-nominal-un	52	318	78.204
cleveland-un	9	746	6.196
cleve-un	4	277	3.848
cloud-un	4	320	0.916
colic-un	7	241	10.316
contraceptiveM-un			
corral-un	6	12	0.076
dermatology-un	10	79	2.172
diabetes-un	4	768	55.068
ecoli-un	9	673	5.248
features_10_train	167	1143	548.544
flags-un	4	10	0.524
glass2-un	2	137	0.964
glass-un	5	688	3.34
haberman-un	13	408	10.684
hayes-roth-un	17	74	0.536
heart-c-un	4	278	3.78
heart-h-un	6	256	5.572
heart-statlog-un	4	266	3.288
hepatitis-un	4	112	1.66
house-votes-84-un	20	85	54.02
hungarian-un	6	300	5.864
irish-un	2	11	0.38
iris-un	5	119	0.496
liver-disorder-un	6	369	6.788
lupus-un	4	128	0.428
lymphography-un	12	65	3.0
molecular-biology_promoters-un	2	92	0.292
mux6-un	8	24	0.096
new-thyroid-un	4	217	1.676
postoperative-patient-data-un	20	107	85.74
promoters-un	2	92	0.28
schizo-un	2	86	6.432
shuttleM-un	5	269	325.296
soybean-un	38	281	19.056
spect-un			
tae-un	10	282	1.16
titanic-un	4	7	0.06
uci_mammo_data-un	18	81	4.164
zoo-un	9	22	0.096

Table A.7: MinDS - SAT with A10 Benchmarks

A.8 MinDS - MinDS$_1$

File	Rules	Literals	Time (seconds)
appendicitis-un			
australian-un			
auto-un			
backache-un			
balance-un			
biomed-un			
breast-cancer-un			
bupa-un			
cars-un			
cleveland-nominal-un			
cleveland-un			
cleve-un			
cloud-un			
colic-un			
contraceptiveM-un			
corral-un	6	16	0.136
dermatology-un			
diabetes-un			
ecoli-un			
features_10_train			
flags-un	5	180	1.552
glass2-un			
glass-un			
haberman-un			
hayes-roth-un			
heart-c-un			
heart-h-un			
heart-statlog-un			
hepatitis-un			
house-votes-84-un			
hungarian-un			
irish-un	4	75	2.784
iris-un			
liver-disorder-un			
lupus-un			
lymphography-un			
molecular-biology_promoters-un			
mux6-un	8	24	0.26
new-thyroid-un			
postoperative-patient-data-un			
promoters-un			
schizo-un			
shuttleM-un			
soybean-un			
spect-un			
tae-un			
titanic-un	4	12	0.064
uci_mammo_data-un			
zoo-un	9	88	6.264

Table A.8: MinDS – MinDS$_1$ Benchmarks

A.9 MinDS - MinDS$_1$ with A10

File	Rules	Literals	Time(seconds)
appendicitis-un			
australian-un			
auto-un			
backache-un			
balance-un			
biomed-un			
breast-cancer-un			
bupa-un			
cars-un			
cleveland-nominal-un			
cleveland-un			
cleve-un			
cloud-un			
colic-un			
contraceptiveM-un			
corral-un	6	12	0.152
dermatology-un			
diabetes-un			
ecoli-un			
features_10_train			
flags-un	5	12	2.272
glass2-un			
glass-un			
haberman-un			
hayes-roth-un			
heart-c-un			
heart-h-un			
heart-statlog-un			
hepatitis-un			
house-votes-84-un			
hungarian-un			
irish-un	4	7	3.472
iris-un			
liver-disorder-un			
lupus-un			
lymphography-un			
molecular-biology_promoters-un			
mux6-un	8	24	0.296
new-thyroid-un			
postoperative-patient-data-un			
promoters-un			
schizo-un			
shuttleM-un			
soybean-un			
spect-un			
tae-un			
titanic-un	4	7	0.076
uci_mammo_data-un			
zoo-un	9	46	6.984

Table A.9: MinDS - MinDS$_1$ with A10 Benchmarks

A.10 MinDS - SMT with Z3 solver

File	Rules	Literals	Time(seconds)
appendicitis-un			
australian-un			
auto-un			
backache-un			
balance-un			
biomed-un			
breast-cancer-un			
bupa-un			
cars-un			
cleveland-nominal-un			
cleveland-un			
cleve-un			
cloud-un			
colic-un			
contraceptiveM-un			
corral-un	6	15	1.164
dermatology-un	11	217	430.6
diabetes-un			
ecoli-un			
features_10_train			
flags-un	4	59	5.444
glass2-un			
glass-un			
haberman-un			
hayes-roth-un			
heart-c-un			
heart-h-un			
heart-statlog-un			
hepatitis-un			
house-votes-84-un			
hungarian-un			
irish-un	8	25	10.296
iris-un			
liver-disorder-un			
lupus-un			
lymphography-un			
molecular-biology_promoters-un	5	42	315.0
mux6-un	8	24	2.184
new-thyroid-un			
postoperative-patient-data-un			
promoters-un	5	42	304.896
schizo-un			
shuttleM-un			
soybean-un			
spect-un			
tae-un			
titanic-un			
uci_mammo_data-un	4	7	0.532
zoo-un	9	38	19.272

Table A.10: MinDS - SMT (Z3 Solver) Benchmarks

A.11 MinDS - SMT with Yices2 solver

File	Rules	Literals	Time(seconds)
appendicitis-un			
australian-un			
auto-un			
backache-un			
balance-un			
biomed-un			
breast-cancer-un			
bupa-un			
cars-un			
cleveland-nominal-un			
cleveland-un			
cleve-un			
cloud-un			
colic-un			
contraceptiveM-un			
corral-un	6	15	0.94
dermatology-un	11	184	548.56
diabetes-un			
ecoli-un			
features_10_train			
flags-un	4	54	5.98
glass2-un			
glass-un			
haberman-un			
hayes-roth-un			
heart-c-un			
heart-h-un			
heart-statlog-un			
hepatitis-un			
house-votes-84-un			
hungarian-un			
irish-un	8	25	9.3
iris-un			
liver-disorder-un			
lupus-un			
lymphography-un			
molecular-biology_promoters-un	5	41	41.02
mux6-un	8	24	1.55
new-thyroid-un			
postoperative-patient-data-un			
promoters-un	5	41	40.76
schizo-un			
shuttleM-un			
soybean-un			
spect-un			
tae-un			
titanic-un			
uci_mammo_data-un	4	9	0.3
zoo-un	9	91	3.84

Table A.11: MinDS - SMT (Yices2 Solver) Benchmarks